

On Efficient Center-based Clustering: From Unsupervised Learning to Clustering under Weak Supervision



By

Avisek Gupta

Electronics and Communication Sciences Unit
Indian Statistical Institute

Under the supervision of

Dr. Swagatam Das

Associate Professor

Electronics and Communication Sciences Unit
Indian Statistical Institute

A thesis submitted to the Indian Statistical Institute
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

November, 2021

*To Truth, whose pursuit
Through an honest empirical lens
Reveals an ever-expanding universe.*

Acknowledgements

The first steps in the field of research can seem daunting, given the expansive ocean of connected literature, the fuzzy notions of what can be significant, and the philosophy of science in general which can take time to properly understand. It is with a lot of gratitude I acknowledge my supervisor Dr. Swagatam Das's role in helping me to, as the saying goes, stand on the shoulder of giants. This journey over the past couple of years that has led to the completion of this thesis was only possible due to the valuable advice I received from him.

I'm also grateful for the advice and support I received from the professors and the staff of the Electronics and Communication Sciences Unit, Indian Statistical Unit, Kolkata. I cannot appreciate enough the valuable interactions I have had over the years with all the students at the unit.

I'm thankful to my parents and my sister for their unwavering constant love and support, through the best of times and through the times of cyclones and the pandemic. Thank you to my friends, who always contribute to my mental fortitude and health: Bibo, Aparajita, Aishani, Sankar, Bibhuti, Sarbendu, Rakesh, Faizan, Sankha, Basma. I can't thank you all enough.

Abstract

The problem of clustering aims to partition unlabeled data so as to reveal the natural affinities between data instances. Modern learning algorithms need to be designed to be applicable on larger datasets that can also be high dimensional. While acquiring more features and instances can be beneficial, the addition of noisy and irrelevant features can also obfuscate the true structure of the data; distance metrics can also fail at high dimensions. To address these challenges, complex mathematical structures can be used to model different aspects of a problem, however they can also lead to algorithms with high computation costs, making the algorithms infeasible for larger datasets.

Among existing classes of clustering methods, we focus on the class of center-based clustering which in general consists of methods with low computation costs that scale linearly with the size of datasets. We identify different factors that have influence over how effective center-based clustering methods can be. Estimating the number of clusters is still a challenge, for which we study existing approaches that have a wide range of computation costs, and propose two low-cost approaches based on two possible definitions of a cluster. Selecting a suitable distance metric for clustering is also an important factor. We incorporate a kernel metric in a center-based clustering method and investigate its performance in the presence of a large number of clusters. Feature selection and feature extraction methods exist to identify which features can help estimate the clusters. We focus on sparse clustering methods and propose a significantly lower computation approach to simultaneously select features while clustering. Another important factor is the nature of the clusters identified. Hard clustering methods identify discrete clusters, whereas soft clustering methods allow soft cluster assignments of data points to more than one cluster, thereby allowing overlapped clusters to be identified. We propose a multi-objective evolutionary fuzzy clustering method that can identify partitions at different degrees of overlap.

Clustering in unsupervised conditions can come with a serious limitation. Instead of exploring a wide solution space completely unsupervised, some additional supervision can bias the method to identify clustering solutions that better fit a dataset. This motivates us to propose a transfer clustering method that learns a multiple kernel metric in a weakly supervised setting, and then transfers the learned metric to cluster a dataset in an unsupervised manner. A lower effort is required to provide weak supervision in comparison to full supervision, while drastically boosting clustering performance. We recommend weakly supervised clustering as a promising new direction to overcome the inherent limitations of identifying clusters in an unsupervised manner.

Contents

1	Introduction to Center-Based Clustering	1
1.1	Data Clustering	2
1.2	Center-Based Clustering	3
1.2.1	Estimating the number of clusters	6
1.2.2	Learning or selecting suitable distance metrics	7
1.2.3	Identifying features that reveal cluster structure	8
1.2.4	The nature of clusters for center-based clustering	11
1.3	A general overview of learning under some degree of supervision	12
1.4	Clustering under some degree of supervision	13
1.5	Scope and Organisation of the Thesis	14
2	Fast Automatic Estimation of the Number of Clusters from the Minimum Inter-Center Distance for k-Means Clustering	18
2.1	Introduction	18
2.2	Motivation	19
2.3	Proposed Methods	21
2.4	Framework for Cluster Number Estimation	24
2.5	Experiments and Results	26
2.5.1	Experimental Setup	27
2.5.2	All Constraints are satisfied	27
2.5.3	Violation of Constraint 1	28
2.5.4	Violation of Constraint 2	31
2.5.5	Violation of Constraint 3	33
2.5.6	Real-world data	37
2.6	Discussion	38
3	On the Unification of k-Harmonic Means and Fuzzy c-Means Clustering Problems under Kernelization	39
3.1	Introduction	39
3.2	Background	41
3.2.1	The k -harmonic means problem	41
3.2.2	The kernel k -means problem	42
3.2.3	The kernel fuzzy c -means problem	42
3.2.4	The generalized fuzzy c -means problem	43
3.3	Kernel k -harmonic means (KKHM)	44

3.3.1	A cost function for KKHMM	44
3.3.2	A common algorithm for KKHMM and KFCM	45
3.4	Experiment and Results	46
3.4.1	Datasets	47
3.4.2	Comparison of kernel clustering algorithms	48
3.4.3	Performance on large number of clusters	50
3.5	Discussion	51
4	Improved Efficient Model Selection for Sparse Hard and Fuzzy Center-Based Clustering	53
4.1	Introduction	53
4.2	Related Work	55
4.2.1	Sparse Clustering Framework and Methods	55
4.2.2	Permutation Method to select the degree of sparsity	57
4.3	Proposed Model Selection for Sparse Clustering Models	58
4.3.1	Bayesian Information Criterion for clustering model selection	58
4.3.2	Bayesian Information Criterion for k -Means and Fuzzy c -Means	60
4.3.3	On Computation Complexity	63
4.4	Experiments and Results	63
4.4.1	Effectiveness of global sparse clustering model selection	64
4.4.2	Comparing sparse clustering methods on synthetic datasets	67
4.4.3	Comparing sparse clustering methods on real datasets	68
4.5	Discussion	71
5	Fuzzy Clustering to Identify Clusters at Different Levels of Fuzziness: An Evolutionary Multi-Objective Optimization Approach	72
5.1	Introduction	72
5.2	Fuzzy Clustering using Multi-Objective Optimization	75
5.2.1	Multi-Objective Optimization	75
5.2.2	Objective Functions for Fuzzy Clusters	75
5.2.3	Multi-Objective Optimization Methods for ECM	78
5.2.3.1	ECM-NSGA-II	78
5.2.3.2	ECM-MOEA/D	79
5.2.4	Computation Complexity of ECM	79
5.3	Experiments and Results	80
5.3.1	Synthetic Datasets	81
5.3.2	Real Datasets	83
5.3.3	Comparison of Pareto Fronts	83
5.3.4	Selection of a Suitable Clustering from the Pareto Set	87
5.4	Discussions	93
6	Transfer Clustering using Multiple Kernel Metrics Learned under Multi-Instance Weak Supervision	94
6.1	Introduction	94
6.2	Transfer Learning for Center-Based Clustering	97
6.3	Related Works	98

6.3.1	Multi-Instance Clustering under Weak Supervision	98
6.3.2	Multiple Kernel Clustering	99
6.4	Methodology	100
6.4.1	Multiple Kernel Multi-Instance k-Means clustering	100
6.4.2	Multiple Kernel Transfer Clustering	103
6.5	Experiments and Results	105
6.5.1	Datasets	105
6.5.2	Experiment Protocol	107
6.5.3	Comparison of Clustering Performances	107
6.5.4	Execution Times of Multiple Kernel Clustering Methods	109
6.5.5	Effect of Number of Bags and Bag Size	110
6.5.6	Empirical Convergence of MKMIKM	111
6.6	Discussions	112
7	Conclusion	113
7.1	Contributions of the Thesis	113
7.2	Future Directions of Research	115
	Appendices	117
A	Supplementary for Chapter 2	117
A.1	Cluster Number Identification Methods	117
A.2	Validation of cluster number estimation methods	124
A.3	Results on Real Datasets	125
B	Supplementary for Chapter 4	126
B.1	Proof of Remark 4.1	126
B.2	Comparison of the execution times when using BIC in comparison to the Permutation Method	128
B.3	Cluster Validity Indices for global sparse clustering model selection	129
B.4	Experiment Results on comparison of approaches for sparse clustering models selection	132
B.5	Experiment Results of sparse clustering methods on synthetic data sets	137
B.6	Influence of the sparsity of the dataset	139
C	Supplementary for Chapter 5	141
C.1	Tuning of parameters for ECM	141
C.2	ECM on datasets with different levels of overlap	147
C.3	Multiple Contradictory Objectives of ECM versus a Combined Single Objective	152
	List of Publications	154
	References	155

List of Figures

1.1	k -Means clustering using Lloyd’s algorithm.	4
1.2	Sub-optimal clustering by the Lloyd’s algorithm.	5
1.3	Layout of the thesis	15
2.1	Identifying the number of clusters from the knee-point plot for a dataset with 3 clusters	20
2.2	The knee-point method often fails for data with higher number of clusters	20
2.3	The last significant reduction in d_k occurs after $k^* = 3$ for the data <i>3clusters</i> (Figure 2.1a) and after $k^* = 25$ for the data <i>25clusters</i> (Figure 2.2a).	21
2.4	The Iris dataset can be clustered into (a) 3 clusters of similar size, or (b) 2 well-separated clusters.	22
2.5	Non-overlapping clusters have distance of at least d between them; when the centers converge into the same cluster, the distance is less than $\frac{d}{2}$	23
2.6	Datasets used to validate the selection criteria for the cluster number estimation methods.	24
2.7	The clusters identified by (a) LL and (b) LML for the validation datasets.	26
2.8	Well-separated clusters are generated with standard deviation σ set to 1, and the minimum possible distance between their centers set to 10.	28
2.9	Accuracy of estimating the number of clusters on datasets satisfying all constraints.	28
2.10	Accuracy of estimating the number of clusters on datasets with clusters having slight difference in the number of points.	30
2.11	Accuracy of estimating the number of clusters on datasets with clusters having high difference in the number of points.	31
2.12	Accuracy of estimating the number of clusters on datasets where clusters have different spread of points.	32
2.13	For neighbouring clusters with high disparity in spread, LL tends to identify well-separated clusters, whereas LML identifies clusters of similar size.	32
2.14	Accuracy of estimating the number of clusters on datasets with clusters having slight overlap.	33
2.15	Accuracy of estimating the number of clusters on datasets with clusters having high overlap.	35
3.1	Synthetic Non-linearly Separable Datasets.	40
3.2	Comparison on synthetic dataset BIRCHlike.	49
3.3	Comparison on synthetic dataset BIRCHlike:diffDensities.	49

3.4	Variations in ARI achieved by KKHM (1)	49
3.5	Variations in ARI achieved by KKHM (2)	50
3.6	A randomly generated dataset containing sparse and dense clusters.	51
3.7	Difference in ARI when running KKHM, KFCM, KKM and BIRCH. Plot depicts ARI for 200 randomly chosen datasets for clarity.	51
4.1	Comparison of the execution times between SKM+BIC and SKM+PM, and between SFCM+BIC and SFCM+PM.	64
4.2	Plots of some of the synthetic datasets used in the experiments.	65
5.1	Clustering a dataset using ECM-NSGA-II to identify clusters at different levels of fuzziness.	77
5.2	The difficulty in clustering a dataset with 3 clusters for different levels of fuzziness.	78
5.3	The synthetic <i>proximity</i> datasets.	81
5.4	The synthetic <i>spread</i> datasets.	81
5.5	Comparison of Pareto fronts on the <i>proximity1</i> synthetic dataset	87
6.1	The proposed Multiple Kernel Transfer Clustering (MKTC) method, described in terms of a source task and a target task.	96
6.2	Empirical convergence of MKMIKM observed over benchmark computer vision datasets.	111
B.1	10-dimensional synthetic data to test the influence of data sparsity on the sparse clustering methods.	139
C.1	Data set containing 3 clusters with different levels of overlap	142
C.2	The variation in maximum ARI with variation in the population size (<i>pop</i>) for ECM-NSGA-II	142
C.3	The variation in maximum ARI with variation in the number of fitness evaluations (<i>FE</i>) for ECM-NSGA-II	143
C.4	The variation in maximum ARI with variation in the fraction of population undergoing genetic operation (<i>pool</i>) for ECM-NSGA-II	143
C.5	The variation in maximum ARI with variation in the number of solutions (<i>tour</i>) from which one solution is selected during tournament selection for ECM-NSGA-II	143
C.6	The variation in maximum ARI with variation in the distribution index for crossover (<i>mu</i>) for ECM-NSGA-II	144
C.7	The variation in maximum ARI with variation in the distribution index for mutation (<i>mum</i>) for ECM-NSGA-II	144
C.8	The variation in maximum ARI with variation in the population size (<i>pop</i>) for ECM-MOEA/D	144
C.9	The variation in maximum ARI with variation in the number of fitness evaluations (<i>FE</i>) for ECM-MOEA/D	145
C.10	The variation in maximum ARI with variation in the neighbourhood size (<i>T</i>) for ECM-MOEA/D	145
C.11	The variation in maximum ARI with variation in the mutation parameter <i>F</i> in Differential Evolution for ECM-MOEA/D	145

C.12	The variation in maximum ARI with variation in crossover parameter Cr in Differential Evolution for ECM-MOEA/D	146
C.13	We consider three datasets to demonstrate the working of the proposed ECM, shown from left to right named respectively as <i>data1-well-separated</i> , <i>data2-slight-overlaps</i> , and <i>data3-high-overlaps</i>	147
C.14	For the dataset <i>data1-well-separated</i> with three well-separated clusters, the Pareto front obtained from ECM-NSGA-II shows a wide range of resulting clusterings. The most suitable clustering for this dataset is obtained from the leftmost solution on the Pareto front, and we see clusterings at different levels of fuzziness across the Pareto front.	148
C.15	For the dataset <i>data1-well-separated</i> with three well-separated clusters, the ECM-MOEA/D Pareto front shows a wide range of resulting clusterings at different levels of fuzziness. For this dataset, the most appropriate clustering is obtained from the leftmost solution on the Pareto front.	149
C.16	For the dataset <i>data2-slight-overlaps</i> with three slightly overlapped clusters, ECM-NSGA-II results in a wide range of clustering solutions for different levels of fuzziness, three of which are shown above, going from the most discrete clustering obtained in the leftmost solution to the most overlapped clustering solution observed in the rightmost solution. The most suitable clustering can be identified within this range of solutions.	149
C.17	For ECM-MOEA/D we observe a behaviour similar to the case of ECM-NSGA-II, where for the dataset <i>data2-slight-overlaps</i> we obtain a wide range of clustering solutions from the most discrete clusters to the left of the Pareto front, to the most overlapped clusterings obtained in the rightmost solution on the Pareto front.	150
C.18	For the dataset <i>data3-high-overlaps</i> with highly overlapped clusters, ECM-NSGA-II results in a Pareto front from which one can obtain a clustering solution with a level of fuzziness that closely matches the underlying degree of overlap of the data.	150
C.19	For the dataset <i>data3-high-overlaps</i> with highly overlapped clusters, ECM-MOEA/D results in a wide Pareto front of solutions covering a wide range of levels of fuzziness, from which we can identify the most suitable clustering solution whose level of fuzziness closely matches the underlying degree of overlap of the data.	151
C.20	For the dataset <i>data1-well-separated</i> shown in Figure C.13(a), the use of the two contradictory objectives of f_1 and f_2 described in eqns. (C.1a) and (C.1b) enables ECM-NSGA-II to identify a wide Pareto front of clustering solutions at different levels of fuzziness, from which one can identify a clustering that closely matches the underlying degree of overlap of the dataset at hand. In contrast, if the two objectives are combined to form a single objective of $f = f_1 - f_2$ which are not contradicting, an evolutionary optimization algorithm will converge to give all equivalent solutions of a clustering with discrete clusters, as observed on the leftmost side of the Pareto front where ECM also obtains its most discrete clustering solutions.	153

List of Tables

1.1	List of Notable Recent Approaches to Estimate the Number of Clusters	6
1.2	Notable Recent Approaches to Learn Distance Metrics	7
1.3	Recent Approaches on Learning Multiple Kernel Metrics while Clustering . .	8
1.4	List of Notable Recent Works on Sparse Clustering	9
1.5	Notable Recent Works on Subspace Clustering	10
1.6	List of Notable Recent Works on Deep Subspace Clustering	11
1.7	Recent Approaches to Identify Suitable Features	11
1.8	List of Approaches for Clustering under Some Degree of Supervision	14
2.1	Specifications of the Cluster Number Estimation Methods.	25
2.2	The Accuracy for Cluster Number Estimation Methods following the Selection Criteria in Table 2.1 on the Validation Datasets in Figure 2.6.	26
2.3	Summary of the performances of the cluster number estimation methods when all constraints are satisfied.	29
2.4	The Execution Times for the Top Performing Cluster Number Estimation Methods.	29
2.5	Summary of the performances of the cluster number estimation methods when clusters have slight differences in the number of points.	30
2.6	Summary of the performances of the cluster number estimation methods when clusters have high differences in the number of points.	31
2.7	Summary of the performances of the cluster number estimation methods when clusters have different variances.	33
2.8	Summary of the performances of the cluster number estimation methods when clusters slightly overlap (a).	34
2.9	Summary of the performances of the cluster number estimation methods when clusters slightly overlap (b).	35
2.10	Summary of the performances of the cluster number estimation methods when clusters highly overlap (a).	36
2.11	Summary of the performances of the cluster number estimation methods when clusters highly overlap (b).	36
2.12	Specifications of the Real Datasets.	37
2.13	The Average Rank of Estimating \hat{k} from Real Datasets.	37
3.1	Popular Kernel Functions.	41
3.2	Summary of Datasets.	47
3.3	Comparison of Performances using ARI.	48

4.1	Specification of the Compared Cluster Validity Indices.	65
4.2	Results of the 21 Approaches for Sparse k -Means Clustering Model Selection across Different Degrees of Sparsity.	66
4.3	Results of the 21 Approaches for Sparse Fuzzy c -Means Clustering Model Selection across Different Degrees of Sparsity.	67
4.4	Average Rank and Wilcoxon Signed-Rank Test on the Average ARI achieved by the Sparse Hard Clustering Methods on the Synthetic Datasets.	68
4.5	Average rank and Wilcoxon Signed-Rank Test on the Average ARI achieved by the Sparse Fuzzy Clustering Methods on the Synthetic Datasets.	68
4.6	Specifications of the Real Datasets.	69
4.7	Average ARI achieved by the Sparse Hard Clustering Methods on the Real Datasets.	70
4.8	Average ARI achieved by the Sparse Fuzzy Clustering Methods on the Real Datasets.	70
5.1	Information to generate the Synthetic Datasets.	81
5.2	Specifications of the Synthetic Datasets.	82
5.3	Comparison of ARI over Synthetic Datasets.	84
5.4	Specifications of the Real Datasets.	85
5.5	Comparison of ARI over Real Datasets.	85
5.6	Comparison of Schott's Spacing Metric over Synthetic and Real Datasets. . .	86
5.7	Comparison of Pareto Fronts on Real and Synthetic Datasets using Epsilon Indicator	88
5.8	The Selection of a Suitable Trade-off Clustering across Different Datasets. . .	90
5.9	(Contd. from Table VI) The Selection of a Suitable Trade-off Clustering across Different Datasets.	91
5.10	Comparison of ARI over the Artificial and Real Datasets	92
6.1	Specifications of the Computer Vision Datasets.	106
6.2	Specifications of the Multi-Instance Subsets formed for each Dataset.	106
6.3	The Average ARI achieved by the Unsupervised and Transfer Clustering Methods.	108
6.4	The Average ARI obtained by MIKM and MKMIKM for Weakly Supervised Multi-Instance Clustering.	108
6.5	The Average ARI obtained by Unsupervised Multiple Kernel k -Means (us-MKKM) in comparison with MKTC.	109
6.6	The Average Execution Time observed for the Multiple Kernel Clustering Methods.	110
6.7	The Average ARI observed for MKTC as the Total Number of Bags is Increased. .	110
6.8	The Average ARI observed for MKTC as the Maximum Possible Size of a Single Bag is Increased.	111
Tables in the Appendices		117
A.1	The Estimated Number of Clusters for each of the Cluster Number Estimation Methods, on the 12 Validation Data Sets.	125

A.2	The Estimated Number of Clusters on Real-world Data Sets.	125
B.1	Execution Times of SKM+PM and SKM+BIC.	128
B.2	Execution Times of SFCM+PM and SFCM+BIC.	128
B.3	(Part I): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.	133
B.4	(Part II): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.	133
B.5	(Part III): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.	134
B.6	(Part IV): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.	134
B.7	(Part I): Average ARI achieved by the Sparse Model Selection Approaches on Sparse Fuzzy c -Means Clusterings.	135
B.8	(Part II): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.	135
B.9	(Part III): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.	136
B.10	(Part IV): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.	136
B.11	Average ARI achieved on Synthetic Datasets by Sparse Hard Clustering Methods.	137
B.12	Average ARI achieved on Synthetic Datasets by Sparse Fuzzy Clustering Methods.	138
B.13	Average ARI achieved by SKM+GAP and SKM+BIC in the presence of increasing number of noise features.	140
B.14	Average ARI achieved by SFCM+GAP and SFCM+BIC in the presence of increasing number of noise features.	140

Chapter 1

Introduction to Center-Based Clustering

In the quest to understand the universe better, mathematical models are used to explain the behaviour of physical phenomena. *Deterministic models* are used to explain events whose inputs can be used to exactly determine the outcome of the event. Deterministic models are not suited to explain events whose outcomes are perceived to be random, which can occur due to randomness in the nature of the event, or when not all input variables that influence the outcome can be observed. The development of statistical theory allows for the creation of *statistical models* that can model the behaviour of random events or processes. The utility of a statistical model can be twofold: to obtain a mathematical model that explains the behaviour of a random process in terms of the relationship between the input and output variables, or to predict the outputs of a random process given its inputs. To model a random process, a statistical model is selected and fed data from the process so that it eventually learns to approximately simulate the behaviour of the random process; this procedure is called *statistical learning*. Popular categorization of statistical learning describes learning procedures in terms of one of the following three categories:

- In *supervised learning* we provide a model with input variables and target responses, and the model is trained to approximately generate the target responses given the corresponding inputs. If the target response is real-valued, the learning problem is known as *regression*; for categorical target responses the learning problem is called *classification*.
- In *unsupervised learning* we have only input data at hand with no target responses, on which we wish to perform certain learning tasks. Popular unsupervised learning tasks include *clustering*, where the natural groups present in the data are to be identified, and *dimension reduction*, where a low-dimension projection of the data is obtained that approximately captures the nature of the data.
- While there are several types of learning possible in between supervised and unsupervised learning, classical categorizations of learning problems referred only to *semi-supervised learning*, where a small part of the data has target responses associated with it, whereas the rest of the data is unlabeled. In present categorizations, semi-supervised learning is now considered part of a general class of methods that offer some degree of

supervision between fully supervised and unsupervised learning. We discuss different approaches to provide some degree of supervision in detail in Section 1.3.

This thesis focuses on a class of clustering methods called center-based clustering to identify naturally occurring groups in the data. We examine different factors that affect the performance of center-based clustering methods. With some additional supervision, the effectiveness of clustering methods can be enhanced significantly, and therefore we also examine the possibilities of clustering under some degree of supervision. In the next section we present a short introduction to data clustering, followed by a general introduction to center-based clustering in the following section.

1.1 Data Clustering

The problem of data clustering is to group data instances into clusters so that instances in the same cluster are similar to each other in comparison to instances in different clusters. This leads to a natural application of clustering to problems such as document clustering, where documents in a collection are grouped together based on which are similar to each other; an analogous application is for the clustering of a collection of images based on the similarity between images. Other applications include image segmentation where the pixels in an image or certain regions in an image are clustered together to form patches that are homogeneous from the perspective of visual semantics; an additional important task is to cluster gene expression data to identify which of the high-dimensional gene expressions have high affinity to each other. Clustering is also a vital initial step in exploratory data mining, where the naturally occurring similarities between data instances are revealed to be further utilized in other data mining tasks. Clustering can also be used as part of a complex learning task, where it is important to identify data instances in the same cluster so that they are treated in the same way.

Clustering methods can be categorized into different classes, where methods in each class can be beneficial in specific situations. In *center-based clustering*, clusters are represented by cluster prototypes that are estimated from the data; this estimation procedure can often be quite efficient. The class of *hierarchical clustering* methods progressively group together data instances based on a distance metric (agglomerative), or they start with all data instances in the same cluster and progressively partition the clusters of instances till a desired number of clusters is reached (divisive). Clustering methods that define clusters as regions with a high density of data instances form the class of *density-based clustering* methods. *Graph-based clustering* consists of graph-cut methods that obtain connected subgraphs from graphs constructed with data instances as vertices and graph edges as the pairwise similarities between data instances. Clustering solutions can also be searched for by single objective or multi-objective *evolutionary search based* methods. Training different types of neural networks to identify clusters on data projected to lower-dimensional latent spaces form the class of *network based* clustering methods. The class of *sequence clustering* models identifies clusters in sequential data such as in text and speech streams. The definition of these categories helps to provide a basic overview of the different types of clustering methods that exist. Specific clustering methods can often possess the qualities that make them belong to more than one category. Among these categories of clustering methods, the class of center-based clustering holds the advantage of generally being more efficient in comparison to methods from other

classes, making the methods suitable for the clustering of large data sets that are nowadays prevalent across all domains of applications.

1.2 Center-Based Clustering

The class of center-based clustering problems usually represents each cluster by a single point called the center of the cluster. Center-based clustering methods generally have lower computation costs in comparison to methods from other classes of clustering, since identifying all clusters requires the estimation of only a single center per cluster. Once the cluster centers are estimated, the cluster memberships of data instances can be easily estimated by identifying which cluster center is closest to each data instance, and assigning the instance to that cluster.

k -Means clustering (MacQueen, 1967; Jain, 2010) is the most popular center-based clustering method, due to its simplicity and low computation cost. As a center-based clustering method, k -Means represents each cluster by a single center of the cluster. The k -Means clustering problem aims to identify k clusters by minimizing its objective function defined as the sum of the squared Euclidean distances of data instances to their cluster centers. Solving the k -Means clustering problem using a brute-force algorithm would require evaluating all $\binom{n}{k}$ possible ways of grouping n data points into k clusters, to determine which grouping leads to the minimizing the k -Means objective function. This approach is however infeasible. An approximate algorithm to the k -Means problem called Lloyd's algorithm (Lloyd, 1982) can efficiently reach a local minima in time linear in the size of the data set. The steps of the Lloyd's algorithm are demonstrated in Figure 1.1.

In contrast to the brute-force approach, Lloyd's algorithm can find a clustering in $O(nk)$ time, even though the final clustering it reaches may be sub-optimal, depending on the location of the random initial cluster centers, shown in Figure 1.2. For complex datasets, multiple sub-optimal local minima may exist. However the low computation cost of Lloyd's algorithm makes it possible to run the algorithm multiple times, from which the best local minima reached can be selected.

Even though several other center-based clustering methods exist, k -Means is still ubiquitous as the first clustering method to try out in any given application. A careful examination of the different factors that contribute to the empirical performance of a center-based clustering method may make it possible to provide a better recommendation of which method can be used in a certain domain of application. As an example, we can consider the Lloyd's algorithm described in Figure 1.1 and make the following observations.

- Since the algorithm starts with a random choice of k cluster centers, the number of clusters k needs to be known in advance. This is a strong assumption, which makes the algorithm applicable only to problems where the number of clusters is known.
- A distance metric is required which is used to identify which cluster center is at the closest distance to each of the data instances, shown in Figure 1.1c. The choice of the distance metric for the data at hand can be critical to identify proper clusters in the data.
- The data features that are measured and collected for the problem play a critical role, since all the features together define the space in which the data instances lie, where the

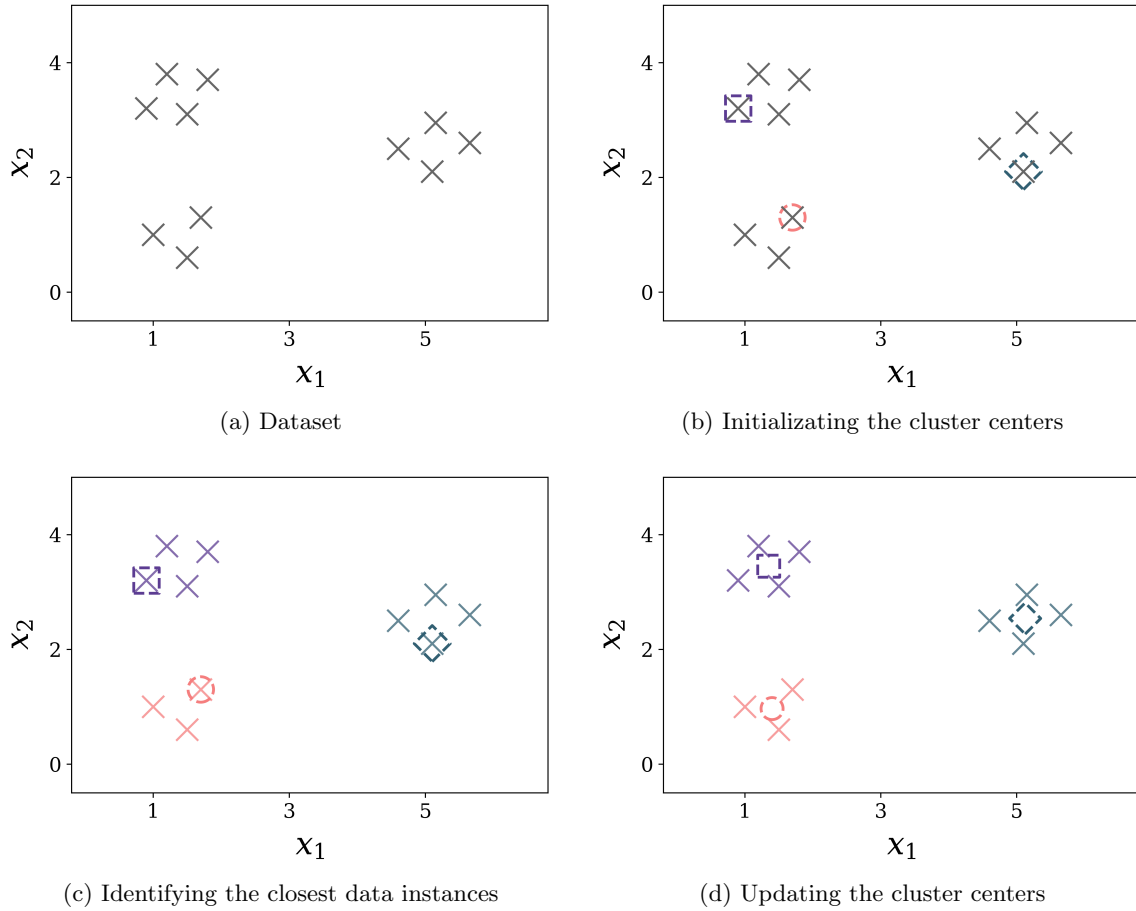


Figure 1.1: k -Means clustering using Lloyd's algorithm. Given an unlabeled dataset shown in 1.1a on which k number of clusters are to be determined, Lloyd's algorithm first randomly initializes k data instances as the initial cluster centers, as shown in 1.1b. Next, the data instances that lie at the closest distance to the cluster centers are identified and assigned to that cluster, shown in 1.1c. Finally, the cluster centers are updated to the location of the mean of all the data instances lying in that cluster, shown in 1.1d. These steps are repeated till the algorithm converges.

distance metric should be defined, and where the clusters are to be identified. Often for many domains of application, several features are collected with the objective of obtaining as much information as possible. However not all features may help identify clusters, due to which a more careful identification of which features should be considered may be necessary.

- Finally, we observe that Lloyd's heuristic assigns each data instance to a single cluster, based on which cluster center lies closest to it. There are variations to this approach, as an example, the Fuzzy c -Means algorithm (Dunn, 1973; Bezdek, 1973; Bezdek et al., 1984) uses fuzzy set theory to make it possible for data instances to have a certain degree of membership to all the clusters under consideration.

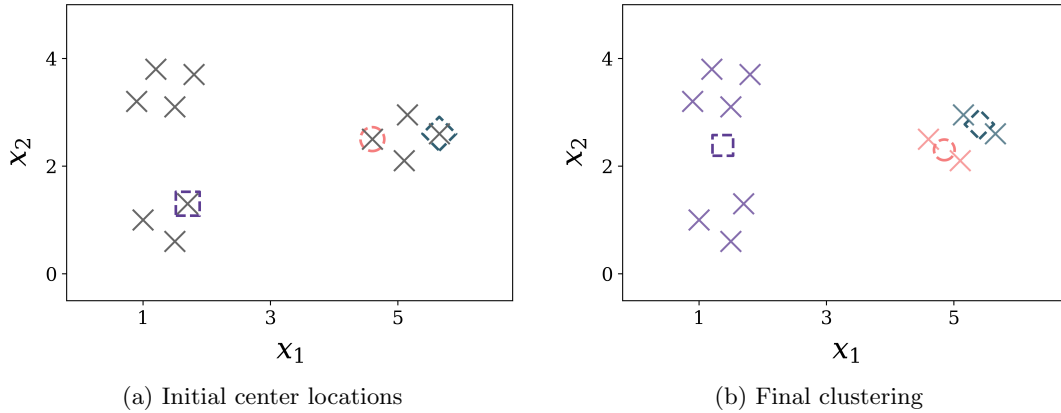


Figure 1.2: The Lloyd's algorithm can converge to a sub-optimal clustering. Given the same dataset of Figure 1.1a, the choice of the initial centers shown in 1.2a can affect the quality of the final clustering reached by the algorithm, shown in 1.2b.

As the above discussion shows, there are different factors that contribute to the performance of center-based clustering methods. An examination of these factors may also allow us to propose ways to improve their performances while also keeping the computation costs low. The following four factors are identified as factors that influence unsupervised center-based clustering methods:

- The *number of clusters* to be identified: The number of clusters to be identified is usually predetermined for popular center-based clusters like k -Means. Learning a suitable number of clusters through a procedure that has low computation complexity is still a challenge. There are higher cost methods that automatically learn it as part of the cluster estimation process, and there are several indices that can be used to check if a certain cluster number is appropriate. However, an estimation method with an overall low cost is still quite desirable.
- The *distance metric* used: The distance metric used can be predetermined by an expert with experience in a specific problem domain. An alternative is to design a method that would automatically learn a distance metric that is best suited to identify clusters in the feature space.
- The *features* used to identify clusters: For modern datasets a wide range of features are often collected in order to obtain more information on the task at hand. When a high number of features are collected, they may not necessarily help identify the underlying cluster structure in the data, as some of the features may be noisy or misleading. This makes those methods advantageous that automatically learn which features to select or drop, or take a linear or non-linear projection of the data that allows for better cluster identification.
- The *nature of clusters* identified: Hard clusters identify discrete clusters where each data instance belongs to only one of the clusters. In comparison, soft clusters (rough or fuzzy) can be fit to identify overlapping clusters, by allowing instances to belong to more than one cluster.

Each of these factors are examined in more detail next, with discussions on recent works to highlight the progress that has been made to address each of the factors.

1.2.1 Estimating the number of clusters

For center-based clustering methods, the number of clusters is often predetermined. Methods from other classes of clustering can often automatically identify an appropriate number of clusters while incurring a higher computation cost. For example, hierarchical clustering methods can repeatedly split clusters or merge instances into clusters until some criteria is met, thereby deciding on the number of clusters; this procedure however requires the computation of all pairwise distances. Evolutionary search based clustering methods can often automatically identify the number of clusters, with the caveat that an evolutionary search across the space of possible clustering solutions requires more computation than center-based clustering methods which often have a more focused estimation procedure involving the alternating optimization of the model parameters.

The general approach to estimate the number of clusters for center-based clustering methods is to obtain clusterings for a range of candidate cluster numbers, and use internal cluster validation methods, or measures of cluster stability, or statistical tests to identify an appropriate number of clusters. Internal cluster validation methods involve defining a notion of an ideal clustering, and evaluating how close the estimated clustering is to it. Cluster stability measures identify whether the same clusters are identified when different sets of instances are drawn from the same data distribution. Cluster number estimation methods using statistical tests usually consider as the null hypothesis a criteria of when a set of instances do not belong to two different clusters, and test for when the null hypothesis can be rejected. Table 1.1 shows some notable recent methods, indices, stability measures or statistical tests to estimate the number of clusters.

Table 1.1: List of Notable Recent Approaches to Estimate the Number of Clusters

Recent Works	Comments
Petrosyan and Proutiere (2016)	Proposed a clustering algorithm that automatically identifies the number of clusters using two components: the first groups neighboring samples together, and the second spreads samples to different clusters.
Gorzalczany and Rudziński (2018)	A generalization of self-organising maps with one-dimension neuron chains that can disconnect and reconnect into subchains to estimate the number of clusters.
Rathore et al. (2019)	Six indices approximating Dunn’s Index at lower computational costs were proposed, where four were based on maximin sampling and two were based on unsupervised training of one class support vector machines.
Srivastava et al. (2019)	Estimated the natural number of clusters in terms of the maximum over two-norms of all the cluster covariance matrices at two successive number of clusters.
Efimov et al. (2019)	Measured homogeneity between two neighboring local clusters using statistical tests of no gaps existing between them, at increasing scales, to identify a clustering and the number of clusters.
Saha and Mukherjee (2021)	A cluster stability approach to identify the number of clusters using cluster centers that involved repetitive sampling data of sizes that were a small fraction of the total dataset size.

1.2.2 Learning or selecting suitable distance metrics

The definition of clusters is based on notions of similarity between instances, therefore the choice of the distance metric used is important. Specific distance or similarity metrics can be well suited for particular domains of applications. They can either be selected, based on an expert’s experience in the domain and the relevant literature on past investigations, or a suitable distance metric can be learnt during the clustering process. Learning a distance metric can involve estimating parameters of the distance metric from the data. For example, the covariance matrix of the Mahalanobis distance, or parameters of a kernel similarity metric can be learnt from the data while simultaneously identifying clusters. Learning a distance metric can also involve selecting a combination of distance metrics. An example of this type of distance learning is learning multiple kernel similarities, where a similarity metric is learnt that is either equal to or is close to a linear combination of predetermined kernel similarities. Table 1.2 provides a list of some notable recent approaches to learning a distance metric or using a new distance metric for a specific clustering task. As can be observed from Table 1.2, recent approaches fall into one of two categories: (i) identifying a distance metric well suited for a particular clustering task; and (ii) estimating parameters of a distance metric while clustering.

Table 1.2: Notable Recent Approaches to Learn Distance Metrics

Recent Works	Comments
Sui et al. (2018)	Proposed a convex clustering objective to learn a Mahalanobis distance metric.
de A.T. de Carvalho et al. (2018)	Learned the Gaussian kernel parameter under two settings where: (i) one parameter is learnt per feature (ii) for each cluster one parameter is learnt per feature.
Marin et al. (2019)	Proved the existence of density biases in the kernel k -Means problems and proposed density equalization methods to resolve them.
Zhang and Cheung (2020)	Defined the distance between data instances and clusters using the order relationships between ordinal data.
Sarkar and Ghosh (2020)	Clustering based on a dissimilarity measure of the mean of absolute differences of pairwise distances, suitable for high dimension low sample sized data.
Vankadara and Ghoshdastidar (2020)	Large sample behaviour for high-dimensional kernel k -Means clustering was analysed, and a subsequent semi-definite relaxation of kernel k -means was proposed.

In Table 1.3 we observe a third category, which involves learning a metric equal to or close to a linear combination of predetermined metrics. There has been recent success in learning a multiple kernel metric best suited for a clustering task, which involves learning a kernel similarity that is equal to or close to a linear combination of predetermined base kernel similarities. This creates a wide search space of possible similarity metrics to search from, in order to select one that best fits the data at hand. Learning multiple kernel metrics has found success in clustering tasks where large base kernel matrices have already been precomputed, and a combined kernel similarity must be learnt while clustering. These base kernel matrices can be large as they contain the pairwise similarities between most if not all data instances. Due to their large size, there can be base kernel matrices with noisy or missing entries, or entire rows and columns can be missing due to certain instances not being present when initially computing the kernel matrix. Recent developments have targeted such challenging situations as well.

Table 1.3: Recent Approaches on Learning Multiple Kernel Metrics while Clustering

Recent Works	Comments
Liu et al. (2016)	Included a regularizer based on pairwise kernel matrix correlation that reduced the contribution of mutually redundant kernels.
Li et al. (2016)	Aligned the combined kernel similarity matrix with the similarity of a sample and its k -nearest neighbours.
Liu et al. (2017b)	Relaxed the search space for the combined kernel matrix to be able to obtain matrices close to the space of a linear combination of base kernels.
Wang et al. (2017)	Provided a framework to optimize for the combined kernel matrix in the presence of outliers in the base kernels.
Kang et al. (2017)	A multiple kernel similarity matrix was learnt with a rank constraint imposed on the Laplacian of the matrix.
Zhu et al. (2018)	Extended Liu et al. (2016) to be able to handle kernels with missing entries.
Nguyen et al. (2018)	Simultaneous clustering while learning a multiple kernel metric, to improve on classification performance.
Han et al. (2018a)	Local kernel weights were combined into a matrix with constraints enforced by an ℓ_p norm, with proved performance bounds.
Kang et al. (2018)	Learned a multiple kernel metric by assigning higher weights to kernel matrices close to a consensus kernel matrix.
Liu et al. (2019)	Clustered incomplete base kernels from which a consensus clustering was obtained.
Liu et al. (2019)	Imputed clustering matrices obtained on base kernels to form a consensus clustering matrix starting from initial zero-filled multiple kernel clustering solutions.
Liu et al. (2020b)	Imputed clustering matrices obtained on base kernels to form a consensus clustering matrix in the neighborhood of a zero-filled multiple kernel clustering solution.
Liu et al. (2020a)	Extended Liu et al. (2017b) to incorporate local density in terms of variable number of neighbors around each instance.
Liu et al. (2020b)	Performed mutual imputation of the multiple kernel matrices while clustering.
Liu et al. (2020a)	Instead of imputation, margins were defined in the space of observed channels of each sample, and the minimum of all sample-based margins was maximized.
Zhou et al. (2020)	Clustering based on neighbor kernels, where neighbors were defined based on similarities in an average kernel matrix.
Yao et al. (2020)	The dissimilarities between kernel matrices were used to identify representative kernels used for clustering.
Ren and Sun (2020)	Preserved global and local structure of data in kernel space, using a kernel self-expressive term and a local structure learning term.

1.2.3 Identifying features that reveal cluster structure

The two prevalent approaches to identify features suitable for clustering are *feature selection* and *feature extraction* approaches. The purpose of feature selection is to identify only those features that are appropriate for clustering, and to drop the rest of the features either because they are close to constant features, or they can be considered to be noisy features, or they do not help to identify the cluster structures in the data. When high-dimensional data is to be clustered, feature selection can help to alleviate the failure of distance metrics in high dimensions. Currently the most popular approach to feature selection for center-based clustering is Sparse k -Means (Witten and Tibshirani, 2010), where an objective function was formulated so as to allow for the sparse optimization of the feature weights, which lead

to certain features being dropped by assigning zero weights to them, whereas the rest of the features were considered to be selected if they were assigned non-zero weights. This method has influenced and motivated subsequent works on simultaneous feature selection and clustering. Some recent notable works on sparse clustering are listed in Table 1.4.

Table 1.4: List of Notable Recent Works on Sparse Clustering

Recent Works	Comments
Azizyan et al. (2015)	Provided theoretical guarantees on clustering performance and feature selection while clustering a mixture of two high dimensional non-spherical Gaussians.
Flammarion et al. (2017)	Obtained sparse clustering formulations with convex relaxations.
Chang et al. (2017a)	Formulated Sparse Fuzzy c-Means with ℓ_q -norm regularization, where $(0 < q < 1)$.
Dey et al. (2020)	Minimized the maximum intra-cluster variance with a sparse regularizer on feature weights.
Chakraborty and Das (2020)	Proposed a strongly consistent lasso weighted sparse k -Means approach to clustering high dimension data.
Chakraborty et al. (2020)	Extended k -Means using generalized power means with entropy regularization to simultaneously learn relevant features while obtaining strongly consistent solutions.
Zhang et al. (2020b)	Proposed a feature ranking based sparse k -Means method with with strong consistency of centers.
Vouros and Vasilaki (2021)	Combined unsupervised feature selection and semi-supervision to create pairwise constraints between instances and identify clusters.

While feature selection aims to retain only those features that help to identify clusters and drop the rest of the features, the objective of feature extraction approaches is to identify lower dimension projections of data that help to reveal the underlying cluster structure. Among feature extraction approaches, subspace clustering ([Vidal, 2011](#)) forms a general class of methods that allows the identification of clusters in subspaces of the original feature space. Sparse subspace clustering (SSC) ([Elhamifar and Vidal, 2013](#)) is an important extension that allowed individual clusters to be identified in different subspaces, without the explicit retrieval of the basis of every subspace. A vast amount of recent work has been done on subspace clustering, of which some notable works are listed in Table 1.5. Subspace clustering is also closely related to spectral clustering approaches, and therefore shares the same limitation of having high computational costs. This has encouraged lower-cost approximation algorithms or sampling based approaches to subspace clustering, where the subspace clustering process works with fewer instances at a time.

Subspace clustering and SSC approaches have high computational costs, while also generally considering only linear projections of data onto different subspaces, where clusters are identified. The recent works on deep subspace clustering address both limitations, by developing lower cost optimization methods to train deep neural networks to learn to project data non-linearly to lower dimension latent spaces where clusters are identified. Table 1.6 lists some important recent works on deep subspace clustering. We observe that often a deep neural network is trained to project data non-linearly to a low-dimensional space, where subspace clustering is performed. The alternative is to train a deep autoencoder, which is a network comprised of an encoder network that projects data non-linearly to a lower dimension latent space where subspace clustering is done, followed by a decoder network that projects the data back to its original space where it verifies that the data instances can be

Table 1.5: Notable Recent Works on Subspace Clustering

Recent Works	Comments
Li et al. (2017)	A single hidden neuron network was trained to generate ℓ_0 sparse codes of data instances on which subspace clustering was performed.
Jalali and Willett (2017)	Subspace clustering that assigned pairwise affinity based on conic geometry, where a strong association was imposed between the tangent cone at an instance and the original subspace containing the instance.
Rahmani and Atia (2017)	Identified subspaces that were novel with respect to other subspaces, by solving a series of linear optimization problems which searched for directions of innovation in the span of the data.
Yu et al. (2018)	The subspace representations of each view were clustered simultaneously while a pairwise co-regularization constraint ensured consistency across views.
Tsakiris and Vidal (2018)	Theoretical guarantees for the sparse subspace clustering of data with missing entries.
Yang (2018)	Obtained ℓ_0 -SSC on data linearly projected onto a lower dimensional space.
Luo et al. (2018)	From multiple views obtained a shared representation consistent with all views and per-view set of specific representations capturing the differences.
Jia and Cheung (2018)	A feature weighting approach to cluster categorical and numerical data with a rival penalized competitive learning scheme that also learned the number of clusters.
Matsushima and Brbic (2019)	Provided theoretical bounds on an efficient SSC method based on selecting instances which most violated the subgradient of the objective function.
Wang et al. (2019c)	Approximation algorithm with theoretical guarantees that applied k -Means to features constructed with rank-restricted Nyström approximation.
Lu et al. (2019)	Guaranteed block diagonal property used by a regularizer to obtain subspace clustering.
Wang et al. (2019)	Identified clusters from multiple statistically independent subspaces while also estimating the number of clusters in each subspace.
Li et al. (2019)	Learned a latent representation close to different views while avoiding using partial information for data reconstruction.
Wang et al. (2019b)	Combined multiple independent self-representations from latent representations of the data.
Liang et al. (2019)	A triplet relationship was used to model the relevance and compactness among three samples to perform subspace clustering while determining the number of clusters.
Zhao et al. (2019)	A regularized Gaussian mixture model based approach that estimated low-dimensional representations of component covariance matrices.
Bai and Liang (2020)	SSC with entropy-norm viewed as spectral clustering with a Gaussian kernel that lowers the computation cost.
Yang et al. (2020)	Subspace clustering by optimizing for sparsity and connectivity by finding mutually strongly connected samples within a subspace.
Chen et al. (2020b)	SSC well suited for larger datasets by using dropouts to operate on a small subset of a dataset.
Kang et al. (2020)	Selected few anchor data instances to construct small anchor graphs per view which were integrated to obtain partitions.
Li et al. (2021)	Sampling-based algorithm that progressively clustered small random samples followed by labeling out-of-sample points.

uniquely reconstructed from its latent space. In addition, in the recent literature we observe multi-view deep subspace clustering methods, which train networks on multiple views or sets of features. We can also observe data augmentation approaches to better learn the subspaces

where clusters can be identified.

From the recent literature on feature selection and extraction for clustering, we also observe novel proposals on projecting data onto lower dimension spaces that have been observed to work well in practice. Some notable works of these kinds are shown in Table 1.7.

Table 1.6: List of Notable Recent Works on Deep Subspace Clustering

Recent Works	Comments
Peng et al. (2017)	Minimized the differences between two centers distributions in the latent space of the deep network, assuming that the distributions are invariant to different distance metrics on the manifold.
Ji et al. (2017)	A self-expressive layer between an autoencoder’s encoder and decoder was proposed to perform subspace clustering.
Zhou et al. (2019)	A distribution consistency loss was used to train an autoencoder to learn a distribution-preserving latent representation.
Mukherjee et al. (2019)	Latent variables sampled from a mixture of one-hot encoded variables and continuous latent variables were used to train data projected to the latent space using a clustering loss
Sun et al. (2019)	Learned latent representations for each of multiple views that were used to learn a common latent subspace.
Abavisani et al. (2020)	Used efficient data augmentation policies to learn consistent subspaces for slightly transformed inputs.
Peng et al. (2020)	A deep neural network was trained for sparse subspace clustering using the l1-norm to enforce sparsity.
Zhang et al. (2020a)	Used complementary information from multiple views to train networks to identify and explore the relationships between latent representations from each view.

Table 1.7: Recent Approaches to Identify Suitable Features

Recent Works	Comments
Shen et al. (2017)	Learned to compress high-dimensional data into short binary codes while also identifying clusters.
Zhang et al. (2017)	Clustering using an $\ell_{2,1}$ -norm constrained matrix factorization with manifold regularizations on low-dimensional feature representations and the cluster indicator matrix.
Liu et al. (2017a)	A low cost approximation algorithm for k -Means clustering on sparse low dimension projections of a dataset.
Yellamraju and Boutin (2018)	Clustering at sequential hierarchical levels based on definitions of clusterability, where at each level binary clusterings are identified on projections of high-dimensional data onto a random one-dimension line.

1.2.4 The nature of clusters for center-based clustering

Among center-based clustering methods, hard clustering methods identify discrete clusters by assigning data instances to only one of the identified clusters. *Soft clustering* methods generalize the notion of cluster memberships to allow data instances to have different degrees of cluster memberships to different clusters. Fuzzy c -Means is perhaps the most popular approach to soft clustering, which defines the possible cluster membership of a data instance

as a fuzzy set to assign soft cluster memberships to multiple clusters (Ruspini et al., 2019; Ruspini, 1969; Dunn, 1973; Bezdek et al., 1984). Possibilistic c -Means Krishnapuram and Keller (1993) is a further generalization of soft clustering that allows low memberships to all clusters for outliers, as well as high membership of instances to more than one cluster. The major advantage of using soft clustering methods is to allow data instances to have different degrees of cluster memberships to all clusters in order to identify overlapped cluster structures in datasets. In real-world datasets it is quite common for clusters to show different degrees of overlap. Using hard clustering methods in such situations may not be suitable, since they will force the partition of the data into discrete partitions. Soft clustering methods can be used instead, since the identification of soft cluster memberships naturally leads to the identification of underlying overlapped cluster structures (Zhou et al., 2020, 2021b; Hu et al., 2021; Zhou et al., 2021a). This appeal of soft clustering methods allows their use across several domains of applications (Zeng et al., 2020; Yang and Benjamin, 2021; Feng et al., 2020; Wu and Zhang, 2021; Bui et al., 2021).

1.3 A general overview of learning under some degree of supervision

Methods that are designed for completely unsupervised learning tasks such as data clustering can be applied to different types of data since unsupervised learning does not require the data to be labeled. In contrast to unsupervised learning, supervised learning methods require all data instances to be labeled, which can be expensive for today’s problems that demand learning algorithms to be trained on datasets of large sizes. In order to reduce the labeling effort required by supervised learning methods, a wide variety of approaches have been proposed that offer different degrees of supervision between the conventional requirements of supervised and unsupervised learning (Ratner et al., 2019 (accessed July 16, 2021); (Ratner et al., 2019, 2020)). Each of these approaches offers some degree of supervision in different ways, while requiring lower labeling cost in comparison to fully supervised learning methods.

Approaches that provide some degree of supervision can be described in terms of four possible categories.

- In *Active Learning*, the learning algorithm actively constructs *queries* to be able to obtain specific supervised information from an expert called the *oracle*. Active learning algorithms generally construct two kinds of queries to the oracle to retrieve information on: (i) the correct cluster label for a data instance, or (ii) whether two data instances lie in the same cluster (known as a must-link constraint), or whether they lie in different clusters (called a cannot-link constraint). Active learning algorithms are designed to minimize the number of queries sent to the oracle, in order to reduce the cost of obtaining labels from experts.
- *Semi-supervised Learning* algorithms usually have access to a smaller labeled dataset and a larger set of unlabeled data. The objective of these learning algorithms is to best utilize both the labeled and the unlabeled data to estimate model parameters so as to best fit the data. The provided supervision can also be in terms of labeled instances, or labeled pairwise constraints, i.e., must-link or cannot link constraints.

- *Transfer Learning* can be described in terms of two tasks: a *source* task and a *target* task. In the source task, a model is trained on a source dataset. In the target task, the entire trained model, or only some of its estimated parameters, are transferred in to accomplish a learning task on a target dataset. In the target task the trained model can be used for the target learning task; the model or some of its parameters can also be part of a larger model that accomplishes the learning task on the target dataset. The target task is usually in a similar domain of application, however recently deep learning models have also achieved great success when transferred to significantly different domains of application.
- *Weak Supervision* focuses on acquiring easier to obtain but potentially noisier labels, often at a higher level of abstraction than that of instance-level labeling. As an example, for the task of image segmentation, instead of labeling every pixel of an image, labels can be provided in terms of what objects are present in the image.

1.4 Clustering under some degree of supervision

Clustering under unsupervised conditions is vastly more popular than clustering under some degree of supervision. There has been less number of investigations on clustering under some degree of supervision, and even fewer works on methods from the class of center-based clustering. We can observe some notable works from the past on general clustering methods in this area, as well as more recent achievements that have been achieved.

In the area of active clustering, optimization methods had been developed to query for either the label of data instances or must-link and cannot-link pairwise constraints from an oracle, so as to minimize the overall number of queries sent to the oracle (Basu et al., 2004a; Grira et al., 2008; Xiong et al., 2013). In the area of semi-supervision, methods had been proposed that used labeled data instances to best estimate the partitions in the underlying feature space, or methods that used pairwise must-link and cannot-link constraints to identify the most appropriate clustering structure (Basu et al., 2006; Kulis et al., 2009; Bair, 2013; Yu et al., 2015; Soares et al., 2017). Transfer clustering approaches had been proposed that utilized some learned parameters or learned models to perform clustering on a task that was different from the original task (Deng et al., 2016; Jiang and Chung, 2012; Han et al., 2019). Weak supervision clustering approaches were focused on using labels that were provided at a higher level of abstraction in comparison to the labeling of data instances directly. This led to weak supervision approaches having an advantage of requiring even less amount of labels in comparison to active clustering and semi-supervised clustering approaches, which can require at most $O(n)$ labels if they are labeling n data instances, or at most $O({}^nC_2)$ labels if they are labeling pairwise constraints. Clustering under weak supervision has also been observed to achieve success in more complex learning tasks such as image cosegmentation (Tao et al., 2017, 2019), where pairs of images are considered at a time to determine the similarity between them, and segment them by clustering the image pixels at the level of superpixels.

Recent works on clustering under some degree of supervision can motivate the future design of center-based clustering methods that operate under some degree of supervision. In Table 1.8 we list some notable recent works on clustering under some degree of supervision.

Table 1.8: List of Approaches for Clustering under Some Degree of Supervision

Recent Works	Comments
Nie et al. (2017)	Simultaneous multi-view clustering or semi-supervised classification along with local manifold structure learning was achieved by modifying the similarity matrix in each iteration until an optimal one is reached.
Xiong et al. (2017)	Active clustering with minimal queries for pairwise constraints by decomposing the expected uncertainty reduction problem into a gradient and a step-scale.
Chang et al. (2017b)	Probabilistic clustering using pairwise constraints from experts while also estimating the experts' accuracies.
Chang et al. (2017c)	A Bayesian probabilistic model that learned multiple clusterings from multiple expert views while assigning higher weights to experts with higher confidence.
Awasthi et al. (2017)	Clustering through user supervised cluster split and merge requests.
Liu et al. (2018)	Obtained partition-level constraints from multiple sources and agents with inconsistencies in the number of clusters to perform clustering as well as saliency-guided co-segmentation.
Wang et al. (2019a)	Semi-supervised clustering where a relative ordering between all instances was maintained based on the must-link and cannot-link constraints.
Kushagra et al. (2019)	Restricted correlation clustering using a sampling procedure with the help of an oracle to sample same-cluster and different-cluster pairs with theoretical performance guarantees.
Bressan et al. (2019)	Active clustering with information-theoretic bounds on the number of queries necessary to guarantee a target disagreement bound.
Li et al. (2020)	Used deep networks to learn embeddings for both labeled and unlabeled instances, while constructing queries based on measures of informativeness and representativeness of instances.
Jing et al. (2020)	Learned a Mahalanobis distance based on must-link and cannot-link constraints provided at a higher level of abstraction.
Shi et al. (2020)	Active clustering that queried to reduce clustering uncertainty, re-labeled related instances, and considered an instance-level weighted voting consensus scheme.
Nie et al. (2020)	Cannot-link graph regularizer with provable guarantee that cannot-link constrained instances are in different clusters.
Tang et al. (2020)	Deep multiple instance learning to identify proposal cluster centers that are spatially adjacent in the same cluster and associated with the same object.
Bai et al. (2020)	A uniform representation for different types of constraints and a consequent clustering method was proposed to identify clusters with high consensus of constraints from multiple sources.
Lai et al. (2021)	Unsupervised base partitions generated from random samples and random subspaces were assigned weights based on internal cluster validation and pairwise constraints, and combined in a consensus approach.

1.5 Scope and Organisation of the Thesis

In this thesis, we study four factors that affect the performance of center-based clustering methods, namely: the number of clusters, the distance measure used, the features selected, and the nature of the clusters identified. We also consider the unsupervised conditions under which clustering methods usually operate to be a limitation, and study possible gains in clustering performance when some degree of supervision is provided. In Figure 1.3 we outline the structure of this thesis following which we discuss each of these topics.

We identify that most center-based clustering methods require the number of clusters to

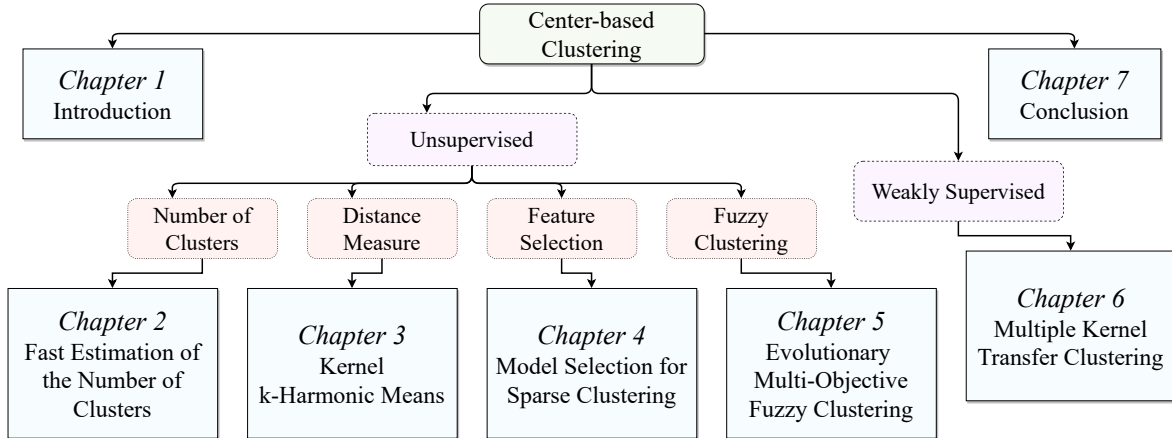


Figure 1.3: Layout of the thesis

be specified. One way to estimate the number of clusters is to consider a range of candidate cluster numbers and use a center-based clustering method to obtain clusterings at each candidate cluster number, followed by using a cluster validation approach to identify the number of clusters. There are several cluster validation methods over a wide range of computation complexity that can be used to identify the number of clusters. We study the overall performance of several popular and recent approaches to identify the number of clusters, and we also propose two methods in Chapter 2 that have low computation costs. The proposed approach is based on our studies of the minimum distance between cluster centers, for which we observe that there are occasional large decreases possible in the minimum distance between cluster centers for successive candidate cluster numbers. We observe that the last time a major decrease occurs can be considered as an appropriate estimate of the number of clusters. We propose two cluster number estimation methods based on this observation, which are proposed corresponding to two possible definitions of a cluster. We conduct extensive studies of our proposed approaches in comparison with several popular and recent cluster number estimation methods, on a number of real and synthetic datasets. In particular, our studies on synthetic generated datasets consider several studies on the behaviour of cluster number estimation methods as the generated datasets deviate away from ideal conditions of clustering.

The second factor that affects center-based clustering methods is the distance measure used. We identify the flexibility that kernel similarities can provide in identifying clusters, and propose an extension of a center-based clustering method called k -Harmonic Means to operate with kernel similarities. In chapter 3 we formulate a kernelized general Fuzzy c -Means objective, which is a general objective from which one can easily derive objectives for Kernel k -Means, Kernel Fuzzy c -Means, and Kernel k -Harmonic Means. We compare the empirical performance of the kernel clustering methods, and in particular study their behaviour in the presence of a large number of clusters, where we observe Kernel k -Harmonic Means to perform significantly better in comparison to the other clustering methods in contention.

The next factor we consider is identifying suitable features for clustering. We focus on center-based clustering methods with sparse regularizers to select features deemed best suited to identify clusters. In particular, we focus on the methods of Sparse k -Means and Sparse

Fuzzy c -Means, and note that both these methods operate by assigning weights to features and imposing an ℓ_1 -norm constraint on the feature weights to induce sparsity by allowing feature weights close to zero to be exactly equal to zero, thus performing feature selection by selecting features with non-zero weights. However the ℓ_1 -norm constraint requires an upper bound, which also allows control over the degree of sparsity allowed, thus controlling the number of features that can be set to zero. Usually a computationally intensive procedure is followed to automatically select the degree of sparsity. In Chapter 4 we derive expressions of the Bayesian Information Criteria for Sparse k -Means and Sparse Fuzzy c -Means to select the degree of sparsity instead, and propose using it instead to select the degree of sparsity. Using the derived expressions lead to significantly lower computation costs. Over extensive experiments we observe that our proposed approaches lead to the identification of better sparse clusterings while incurring lower computation costs.

The fourth factor that affects center-based clustering is the nature of the clusters identified. Fuzzy clustering approaches are a general approach to identify clusters with different degrees of overlap. Usually fuzzy clustering methods like Fuzzy c -Means have a parameter that can be used to control the degree of overlap that is identified between clusters. In Chapter 5 we propose an evolutionary multi-objective fuzzy clustering method that automatically obtains several fuzzy clusterings at different levels of fuzziness corresponding to different degrees of overlap identified between clusters. This is achieved by the use of contradicting objectives in a multi-objective setup, where one of the objectives is a measure of intra-cluster variance that tries to produce compact discrete clusters while the other objective is a measure of the entropy of cluster memberships that tries to produce completely overlapped clusters. Optimizing both in a multi-objective setup produces a Pareto front of a wide range of optimized clusterings at different levels of fuzziness corresponding to different degrees of overlap, from which a suitable clustering for the application at hand can be selected. We also propose a method to identify a clustering from the Pareto front, and evaluate our proposed approach over several real and synthetic datasets.

Along with our studies on the four factors that affect the performance of center-based clustering methods, we look at the unsupervised conditions under which they generally operate as a limitation as well, and expect significant improvements in clustering performances to be possible if some degree of supervision can be provided. Recently, multiple kernel clustering methods have achieved great success in learning a multiple kernel similarity metric while also clustering simultaneously. Multiple kernel clustering methods search for a suitable multiple kernel similarity metric over a wide space of possible similarity metrics. In Chapter 6 we are motivated by the idea that some degree of supervision may help bias a multiple kernel clustering methods to learn suitable multiple kernel metrics, instead of searching for the metric in an unsupervised manner over a wide search space. We propose a method that can be described in terms of two tasks, a source task and a target task. The source task involves learning a multiple kernel metric in a weakly supervised multi-instance setup, where multiple data instances are grouped together and assigned possible labels, which leads to lower labeling efforts. The multiple kernel metric learnt under this weakly supervised setup is transferred to the target task which is an unsupervised single instance clustering task. Our overall method has significantly lower computation costs, even in comparison to the state-of-the-art unsupervised multiple kernel clustering methods. Over several large computer vision datasets we observe that significant improvements are achieved by our proposed method while incurring lower computation costs.

Based on our overall studies and observations on center-based clustering methods, we provide concluding remarks in Chapter 7 along with discussions on interesting subsequent directions of research that are possible, especially in the promising direction of clustering under some degree of supervision.

Chapter 2

Fast Automatic Estimation of the Number of Clusters from the Minimum Inter-Center Distance for k -Means Clustering

Summary

Center-based clustering methods like k -Means intend to identify closely packed clusters of data points by respectively finding the centers of each cluster. However, k -Means requires the user to guess the number of clusters, instead of estimating the same on the run. Hence, the incorporation of an automatic estimation of the natural number of clusters present in a dataset is important to make a clustering method truly unsupervised. For k -Means, the minimum of the pairwise distance between cluster centers decreases as the user-defined number of clusters increases. In this chapter, we observe that the last significant reduction occurs just as the user-defined number surpasses the natural number of clusters. Based on this insight, we propose two techniques: the Last Leap (LL) and the Last Major Leap (LML) to estimate the number of clusters for k -Means. Over a number of challenging situations, we show that LL accurately identifies the number of well-separated clusters, whereas LML identifies the number of equal-sized clusters. Any disparity between the values of LL and LML can thus inform a user about the underlying cluster structures present in the dataset. The proposed techniques are independent of the size of the dataset, making them especially suitable for large datasets. Experiments show that LL and LML perform competitively with the best cluster number estimation techniques while imposing drastically lower computational burdens.

2.1 Introduction

Center-based clustering methods offer a low computation approach to identifying clusters in datasets. This is achieved by representing each cluster by a (usually single) center of the cluster. The estimation of the cluster representatives lowers the computation cost significantly, an example of which was discussed in Figure 1.1 in Chapter 1. Center-based clustering methods commonly require the number of clusters to be provided as input by the

user along with the input data, which raises the question: If center-based clustering methods cannot automatically estimate the natural number of clusters in a dataset, are they truly unsupervised?

Estimating the natural number of clusters is generally not trivial since the underlying data distribution is unknown. Consequently, there exists a large number of different approaches to estimate the number of clusters (Hancer and Karaboga, 2017). Cluster validity indices attempt to identify what properties an ideal clustering should have, and accordingly assign a numeric score to a clustering of data points. Naturally, cluster validity indices have been used previously to estimate the number of clusters (Milligan and Cooper, 1985; Dimitriadou et al., 2002; Arbelaitz et al., 2013). Other than cluster validity indices, there exists a school of diverse methods to estimate the number of clusters (Tibshirani et al., 2001; Tibshirani and Walther, 2005). There are also clustering methods where the estimation of the number of clusters is built into the method itself (Frey and Dueck, 2007; Handl and Knowles, 2007; Louhichi et al., 2017; Du et al., 2017; Liang and Chen, 2016; Tong et al., 2017). There is also the general class of divisive clustering methods, where starting from one or two clusters, the clusters are repeatedly divided until all clusters satisfy a well-defined condition (Savaresi and Boley, 2001; Hamerly and Elkan, 2003). These methods make the strong assumption that the terminal number of clusters is close to the natural number.

Given such a large variety of approaches to estimate the number of clusters across different categories of data clustering methods, comparing all of them in a meaningful way becomes difficult. Therefore, in this chapter we restrict our focus to the center-based k -Means clustering algorithm MacQueen (1967); Jain (2010), for which we investigate which method best estimates the number of clusters. In the following section we discuss our motivations for developing two new cluster number estimation methods: the Last Leap (LL) and the Last Major Leap (LML). Both methods observe only the minimum distance between cluster centers to estimate the cluster number, consequently having computational complexity far lower than existing methods (making them much faster on large datasets). Through a number of experiments we compare our methods with other existing methods for k -Means clustering to determine which accurately estimates the natural number of clusters.

2.2 Motivation

Given a dataset of n points $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$, and a user-defined number of clusters k , k -Means divides the dataset into k clusters C_1, \dots, C_k , where each cluster C_j is represented by a cluster center \mathbf{v}_j . Let k^* denote the natural number of clusters in a dataset. Our objective is to find an estimate \hat{k} for k^* .

Existing cluster number estimation methods often consider two conditions that an ideal clustering should satisfy, to arrive at such an estimate. The two conditions are: (i) *Compactness*, where the cluster around each center is dense, and (ii) *Separation*, where different clusters are far apart.

Compactness: A notion of compactness often exists in cluster validity indices in the form of the sum of the average intra-cluster pairwise-distances W_k , which is also equal to the sum of the squared distances of each point to its nearest cluster center,

$$W_k = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2.$$

W_k decreases with increasing values of k . A plot of W_k can show that at the natural number of clusters, there is a large reduction in the value of W_k . This leads to a basic method to estimate the number of clusters where the *knee-point* of the plot of W_k is identified, as shown in Figure 2.1. The knee-point of this plot can be easily computed by finding the point that lies at a maximum distance to the line joining the first and the last point. However, the knee-point method tends to fail for higher values of k^* , as shown in Figure 2.2.

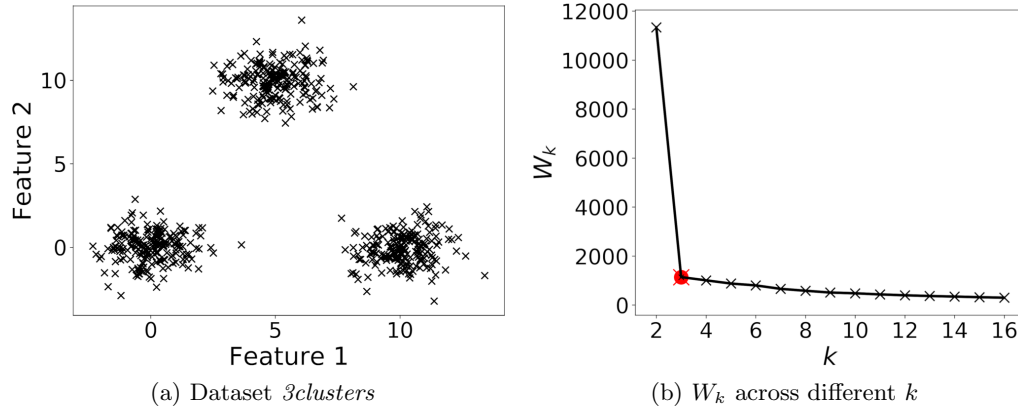


Figure 2.1: The data has 3 clusters, which can be identified from the knee-point in the plot of W_k over increasing values of the number of clusters k .

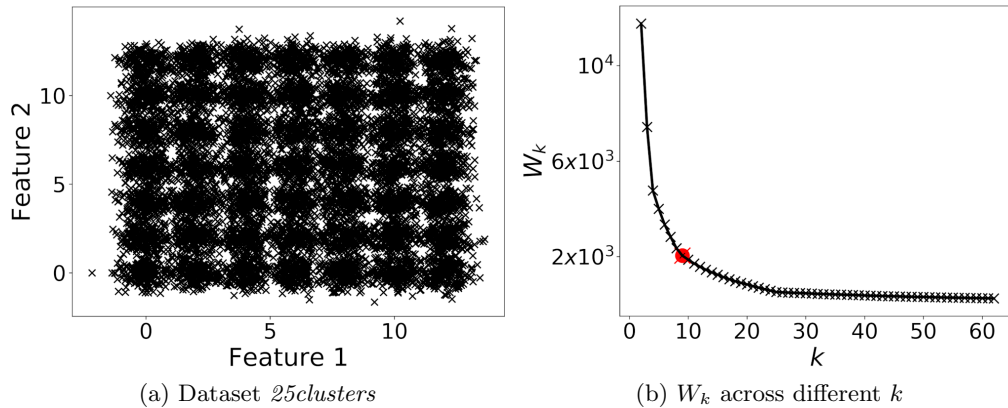


Figure 2.2: The knee-point often fails for higher number of clusters. The dataset has 25 clusters, however the knee-point identifies only 9 clusters.

Separation: Among several approaches to consider separation of clusters, one is to use the minimum distance between cluster centers d_k (Xie and Beni, 1991),

$$d_k = \min_{i \neq j} \|\mathbf{v}_i - \mathbf{v}_j\|^2.$$

The plots of d_k across increasing values of k in Figure 2.3 show an interesting behaviour. While d_k generally decreases as k increases, there are occasional *leaps* in the reduction in d_k . What is noteworthy is that there is always one *final significant leap* that occurs just as k surpasses

k^* . After this last significant leap, the reduction in d_k continues to be gradual. Even when the natural number of clusters is high (> 20), as long as the clusters are well-separated, the final significant leap does occur immediately after the natural number of clusters. This leads to the question of whether this last significant reduction can always be correctly identified. Based on this question, we propose two cluster number estimation methods in the following section.

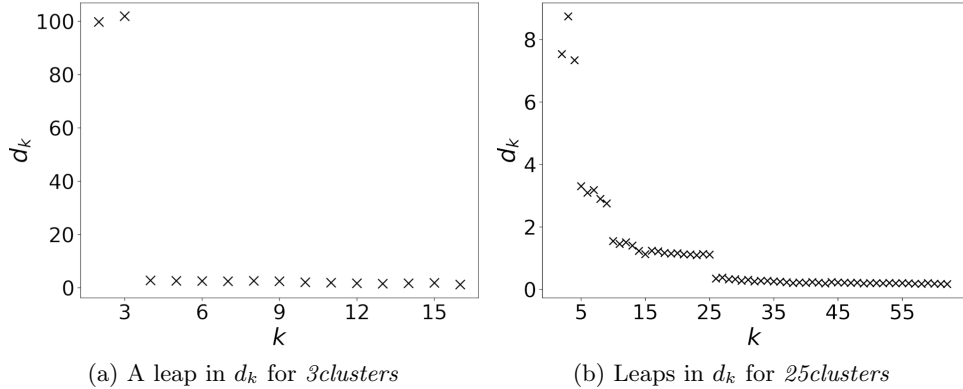


Figure 2.3: The last significant reduction in d_k occurs after $k^* = 3$ for the data *3clusters* (Figure 2.1a) and after $k^* = 25$ for the data *25clusters* (Figure 2.2a).

2.3 Proposed Methods

The primary objective of creating a cluster number estimation method is to accurately estimate the natural number of clusters k^* . However, the natural number of clusters is often difficult to define. Often in ambiguous situations it is hard to reach a consensus on what the value of k^* should be. As a simple example, Figure 2.4a shows the Iris dataset which has three classes. The literature on cluster number estimation often expects clustering methods to identify these three classes as three clusters since they are of somewhat comparable sizes (Fujita et al., 2014; Rezaee et al., 1998; Zhao et al., 2009b). However, Figure 2.4b shows that two of the classes are not well-separated, hence the recognition of two clusters should perhaps also be considered as valid. In situations where such an ambiguity exists regarding the number of clusters, the estimated \hat{k} considered to be the natural number of clusters may depend on the application at hand. Also from the perspective of data exploration, it may be beneficial to know whether multiple valid \hat{k} may exist for a given data. Therefore, we propose two methods to estimate the number of clusters: the *last leap* which is more prone to identifying well-separated clusters, and the *last major leap* which tends to identify clusters of equal size.

1. The Last Leap (LL): We propose the following index to identify where the last significant difference occurs in the values of d_k ,

$$LL(k) = \frac{d_k - d_{k+1}}{d_k}. \quad (2.1)$$

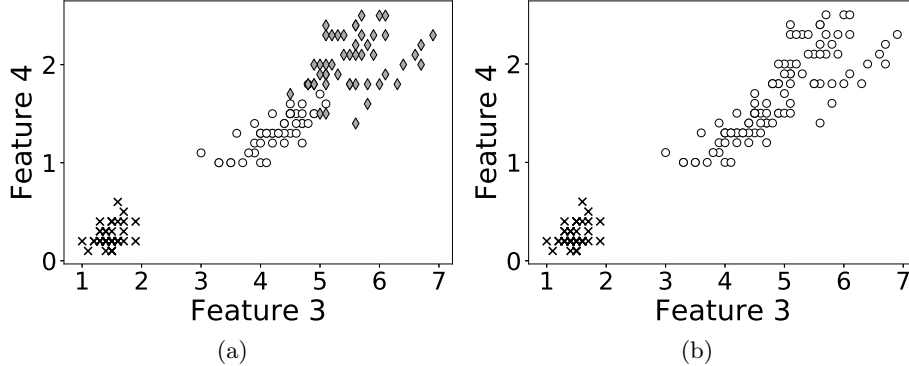


Figure 2.4: The Iris dataset can be clustered into (a) 3 clusters of similar size, or (b) 2 well-separated clusters.

The k at which LL is maximized is defined to be the estimated \hat{k} in the following way:

$$\hat{k}_{LL} = \arg \max_{k=2, \dots, k_{max}} LL(k).$$

A large value of the numerator $d_k - d_{k+1}$ of LL helps to indicate where a large reduction has occurred. The denominator contains d_k which generally decreases with increasing k , thereby helping to identify where the ‘last’ large reduction in LL occurred.

2. The Last Major Leap (LML): We define the method LML to specifically check for the last significant reduction in d_k :

$$I_{LML}(k) = \begin{cases} 1 & , \text{ if } \frac{1}{2}d_k > \max_{l=k+1, \dots, k_{max}} d_l, \\ 0 & , \text{ otherwise,} \end{cases} \quad (2.2)$$

$$K = \{k \mid I_{LML}(k) == 1\}, \text{ and} \quad (2.3)$$

$$\hat{k}_{LML} = \begin{cases} \max K & , \text{ if } K \neq \phi. \\ 1 & , \text{ otherwise.} \end{cases} \quad (2.4)$$

Here $I_{LML}(k)$ is an indicator function that identifies a significant leap when half of the current d_k is greater than d_l , for higher values of l ($l = k+1, \dots, k_{max}$). K forms the set of the number of clusters k at which a significant leap in d_k occurs. From the set K , the largest value is selected as the estimated \hat{k} and is considered to be the last significant leap.

We next justify how I_{LML} identifies significant leaps. The k -Means clustering algorithm identifies spherical crisp clusters of equal size that do not overlap. We assume that the data contains clusters of this nature which is ideal for k -Means. At the natural number of clusters $k = k^*$, k -Means correctly identifies the center of each cluster. The minimum distance between centers d_k can then be the diameter d of the clusters, if any two clusters touch. Otherwise d_k is some value greater than d . Therefore at $k = k^*$, $d_k \geq d$.

At $k = k^* + 1$, k -Means will place two centers in the same cluster, leading to a minimum inter-center distance that is less than $d/2$. Therefore when k is increased from k^* to $k^* + 1$, the decrease in minimum inter-center distance is greater than $d - d/2 = d/2$. I_{LML} is defined to identify this decrease. Furthermore for very high values of k where d_k is close to zero,

possible noisy variations in d_{k+1} can be avoided by considering the maximum over all d_l , $l = k + 1, \dots, k_{max}$.

While such large reductions in d_k may certainly occur between k and $k + 1$ when $k < k^*$, they do not occur when $k > k^*$. For any $k > k^*$, between k and $k + 1$ there are two possibilities for k -Means to place the new center. The center can either be placed in a cluster that previously had only one center. This does not have a substantial effect on d_k , since there already exists another cluster with more than one center in it. The other possibility is that the center is placed in a cluster that already has more than one center. Then d_{k+1} may decrease compared to d_k , but the difference will not be greater than $\frac{1}{2}d_k$.

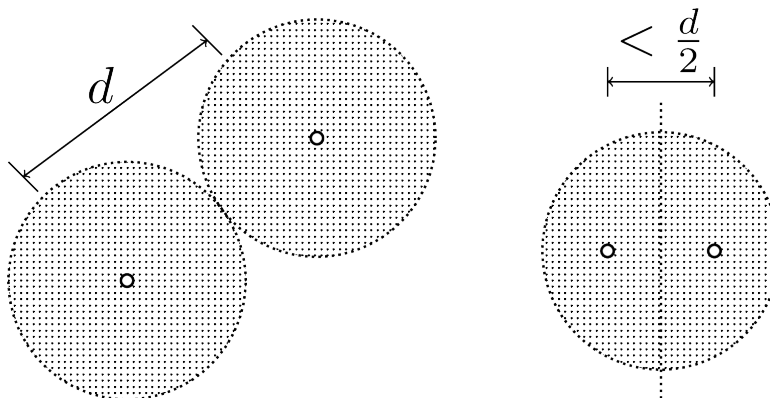


Figure 2.5: The centers of non-overlapping clusters have at least distance d between them, where d is the diameter of the clusters. When two centers converge in the same cluster, the distance between them is less than $\frac{d}{2}$.

On identifying single clusters: In (2.2), I_{LML} may not identify any significant leaps, leading to an empty set K in (2.3). If such a situation arises, we can consider the data contains no clear cluster structures, and therefore in (2.4) \hat{k}_{LML} is set to 1. A similar rule is incorporated to LL as well:

$$\hat{k}_{LL} = 1, \text{ if } \frac{1}{2}d_k < \max_{l=k+1, \dots, k_{max}} d_l. \quad (2.5)$$

For more complex datasets where the natural number of clusters may be more difficult to decide, LL and LML give different results based on two different perspectives. In eqn. (2.4), \hat{k}_{LML} is the largest k for which $I_{LML}(k) = 1$ from eqn. (2.3). This causes LML to prefer the division of a dataset into equal-sized clusters. On the Iris dataset, LML identifies 3 clusters. The numerator in eqn. (2.1) is the difference between d_k and d_{k+1} . Therefore, faced with an ambiguous situation LL will choose a k that identifies well-separated clusters. For the Iris dataset, LL identifies 2 clusters.

Both LL and LML work only on the minimum distance between the cluster centers, and do not require the dataset at all. Both have a computation complexity of only $O(k^2)$. Since usually $k \ll n$, both LL and LML are drastically more efficient than other cluster number estimation methods, especially on large datasets.

2.4 Framework for Cluster Number Estimation

In this section, we state the framework (Milligan and Cooper, 1985; Dimitriadou et al., 2002) followed to compare LL and LML with the popular and state-of-the-art cluster number estimation methods:

1. For $k = 1, 2, \dots, k_{max}$, run k -Means to identify k clusters in the given dataset.
2. Run a cluster number estimation method on all output clusterings for $k = 1, \dots, k_{max}$.
3. Use a suitable selection criteria to estimate the number of clusters.

Along with LL and LML, we evaluate the performance of 28 existing cluster number estimation methods compatible with the stated framework. All methods are listed in Table 2.1. The definitions of each method can be found in Section A.1 of Appendix A.

Each cluster number estimation method compatible with the framework can be written as a function $f_k : \{C_1, \dots, C_k\} \rightarrow \mathbb{R}$, where f_k evaluates a clustering with k clusters. Given clusterings for $k = 1, \dots, k_{max}$, each method therefore returns k_{max} values $f_1, \dots, f_{k_{max}}$. A *selection criteria* associated with each method is used to select the number of clusters from these k_{max} values. The selection criteria is generally one of the following:

1. The maximum or minimum of f_k .

$$\hat{k} = \arg \max_k / \arg \min_k f_k.$$

2. The knee-point method, which selects $\hat{k} = k$ for the f_k lying at maximum distance to the line joining f_1 and $f_{k_{max}}$.
3. A rule specific to the method, such as the 1-standard-error rule followed by the Gap Statistic (Tibshirani et al., 2001).

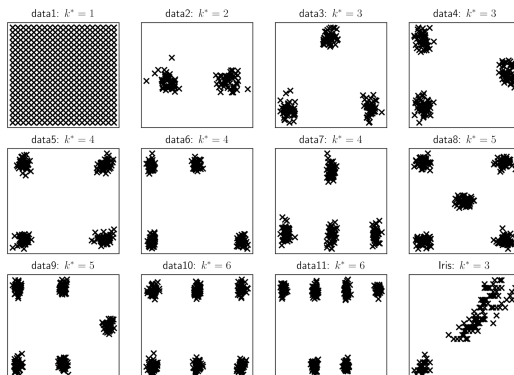


Figure 2.6: Datasets used to validate the selection criteria for the cluster number estimation methods.

Following this categorization, the selection criteria for each method is provided in Table 2.1. In the literature, there are often inconsistencies regarding the selection criteria for

2. Fast Automatic Estimation of the Number of Clusters

Table 2.1: Specifications of the Cluster Number Estimation Methods.

Sl. No.	Name	Complexity	Selection Criteria for \hat{k}	Min no. of clusters
1	Akaike Information Criterion (AIC) (Akaike, 1974)	$O(nk)$	Knee	2
2	Bayesian Information Criterion (BIC) (Schwarz, 1978)	$O(nk)$	max	2
3	Caliński Harabasz (CH) Index (Caliński and Harabasz, 1974)	$O(nk)$	max	2
4	Classification Entropy (CE) (Bezdek, 1975)	$O(nk)$	min	2
5	Compose Within-Between (CWB) (Rezaee et al., 1998)	$O(nk)$	min	2
6	Davies Bouldin (DB) Index (Davies and Bouldin, 1979)	$O(nk)$	Knee	2
7	Dunn Index (Dunn, 1973)	$O(n^2)$	max	2
8	Knee-point Method	$O(k)$	Knee	1
9	Fukuyama Sugeno (FS) Index (Fukuyama and Sugeno, 1989)	$O(nk)$	Knee	1
10	Fuzzy Hypervolume (FHV) (Dave, 1996)	$O(nk)$	min	1
11	Gap Statistic (Tibshirani et al., 2001)	$O(Bnk)$	1-standard-error rule	1
12	Halkidi Vazirgiannis (HV) Index (Halkidi and Vazirgiannis, 2001)	$O(nk^2)$	min	2
13	Hartigan Index (Hart) (Hartigan, 1985)	$O(nk)$	Knee	1
14	I Index (Maulik and Bandyopadhyay, 2002)	$O(nk)$	max	2
15	Jump Method (Sugar and James, 2003)	$O(nk)$	max	2
16	Modified Partition Coefficient (MPC) (Dave, 1996)	$O(nk)$	max	2
17	Partition Coefficient (PC) (Bezdek, 1973)	$O(nk)$	max	2
18	Partition Index (PI) (Bensaid et al., 1996)	$O(nk)$	Knee	2
19	PBMF (Pakhira et al., 2004)	$O(nk)$	max	2
20	PCAES (Wu and Yang, 2005)	$O(nk)$	max	2
21	Prediction Strength (PS) (Tibshirani and Walther, 2005)	$O(n^2)$	Prediction Threshold	1
22	Ren Liu Wang Yi (RLWY) Index (Ren et al., 2016)	$O(nk)$	min	2
23	Silhouette Index (SIL) (Rousseeuw, 1987)	$O(n^2)$	max	2
24	Slope Statistic (Fujita et al., 2014)	$O(n^2)$	max	2
25	Xie Beni Index (XB) (Xie and Beni, 1991)	$O(nk)$	min	2
26	Xu Index (Xu, 1997)	$O(nk)$	min	2
27	Zhao Xu Fränti (ZXF) Index (Zhao et al., 2009b)	$O(nk)$	Knee	2
28	SC Index (Rezaee, 2010)	$O(nk^2)$	min	2
29	Last leap (LL)	$O(k^2)$	max & (2.5)	1
30	Last Major Leap (LML)	$O(k^2)$	(2.3) & (2.4)	1

several methods. Therefore we used a set of 12 datasets (shown in Figure 2.6) to validate the selection criteria. The number of clusters spans from 1 to 6, with the position of the clusters changing to check if that affects the performance of the estimation method. The accuracy of each method in correctly identifying the number of clusters is shown in Table 2.2. We also show the performance of LL and LML on these datasets in Figure 2.7. Both have estimated the correct number of clusters for all but Iris, where LL estimates 2 clusters and LML estimates 3. Section A.2 of Appendix A provides the complete results of the performance on the validation datasets. In Table 2.1, we also report the minimum number of clusters each method can identify as well as the computation complexity of the methods.

Table 2.2: The Accuracy for Cluster Number Estimation Methods following the Selection Criteria in Table 2.1 on the Validation Datasets in Figure 2.6.

Method	Accuracy (%)	Method	Accuracy (%)	Method	Accuracy (%)
AIC	75.00	Gap	83.33	PBMF	91.67
BIC	83.33	HV	83.33	PCAES	83.33
CH	91.67	Hart	41.67	PS	83.33
CE	83.33	I	91.67	RLWY	66.67
CWB	83.33	Jump	91.67	Rez	75
DB	75.00	LL	91.67	SIL	83.33
DUNN	75.00	LML	100	Slope	83.33
Knee	83.33	MPC	83.33	XB	83.33
FS	66.67	PC	83.33	Xu	83.33
FH	91.67	PI	75	ZXF	75

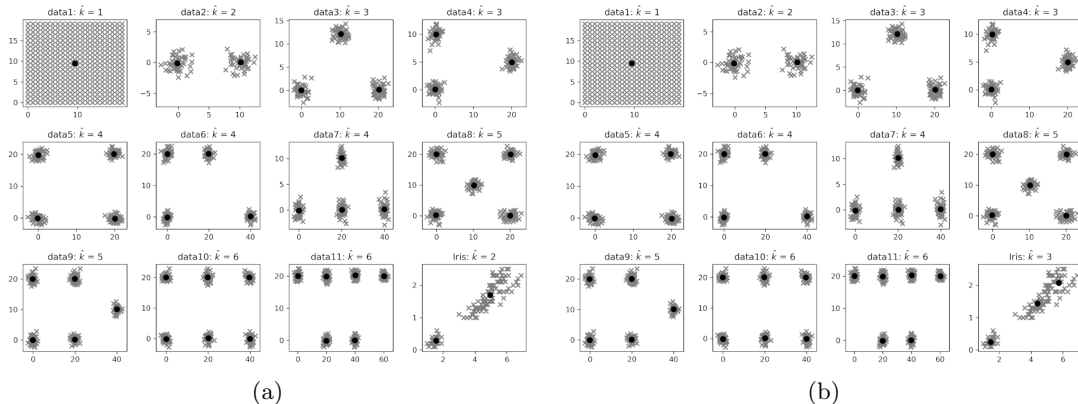


Figure 2.7: The clusters identified by (a) LL and (b) LML for the validation datasets.

2.5 Experiments and Results

In this section, we first investigate the performance of all cluster number estimation methods under conditions ideal for k -Means. k -Means fits a dataset with k spherical multivariate Gaussian distributions, having equal variance along all dimensions. Hence k -Means works best on datasets satisfying the following constraints:

Constraint 1: *All clusters have equal number of points.*

Constraint 2: *All clusters have equal variance.*

Constraint 3: *All clusters are well-separated.*

In the first experiment we investigate the performance of all cluster number estimation methods under ideal conditions, where all constraints are satisfied. This is followed by investigations on how the performances of the methods drop when each constraint is individually violated. These deviations away from the ideal conditions give us an estimate of the performance of each method on more realistic scenarios. Finally, we evaluate the performance on real-world datasets. Before we discuss the results from each experiment, we first discuss the experiment protocol maintained throughout the study.¹

2.5.1 Experimental Setup

For each experiment, random datasets are generated from Monte Carlo simulations, maintaining the necessary conditions discussed previously. For each dataset, 100 data points are drawn from each cluster having variance 1 along each dimension. When the variance is fixed at 1, data generated around the origin mostly lie in the range of $[-4, 4]$. Therefore, the minimum distance between the cluster centers is set to 10 to ensure that the clusters are well-separated. For each dataset of size n , k -Means with k -Means++ initial centers (Arthur and Vassilvitskii, 2007) is run 30 times for each value of k from 1 to $k_{max} = \sqrt{n}$, for a maximum of 300 iterations, with a termination threshold of 10^{-16} . The clustering yielding the lowest squared error from the 30 runs of k -Means is selected. For methods such as the Gap Statistic or the Prediction Strength that require further clusterings of reference sets or subsets of the data, the number of runs of k -Means is also set to 30. We compute the accuracy of each method as the percentage of datasets for which it correctly identifies the number of clusters. In addition, the results of the experiments are summarised by the average rank obtained by each method across all datasets, in addition to which we conduct the Epps-Singleton test to test for statistical significance of the results obtained by the proposed methods in comparison to all other methods in contention.

2.5.2 All Constraints are satisfied

In this experiment we investigate the performance of all cluster number estimation methods under ideal conditions. We wish to test the performances for increasing number of dimensions as well as increasing number of clusters. Therefore we generate datasets of dimensions 2, 10, 20, 35 and 50. For each dimension, 2, 10, 20, 35 and 50 number of clusters are generated. In this manner, 25 datasets can be generated. We generate 50 groups of data, each group containing 25 random datasets, over which the average accuracy is measured.

The accuracy in estimating the number of clusters for all the methods is shown in Figure 2.9. We observe LL, LML, BIC, CH, CE, FHV, I, MPC and PBMF obtain the highest accuracy, followed closely by Dunn, PC, Slope and Xu. Considering only the minimum distance between centers has thus led both LML and LL to be among the top-performing

¹The source codes for LL, LML, and all the experiments are available at https://github.com/Avisek20/cluster_number_estimation.

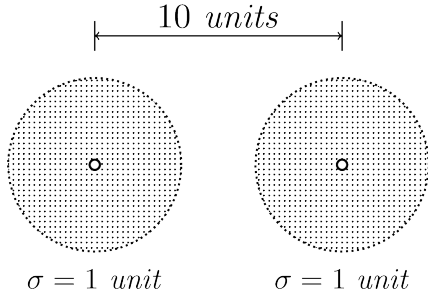


Figure 2.8: Well-separated clusters are generated with standard deviation σ set to 1, and the minimum possible distance between their centers set to 10.

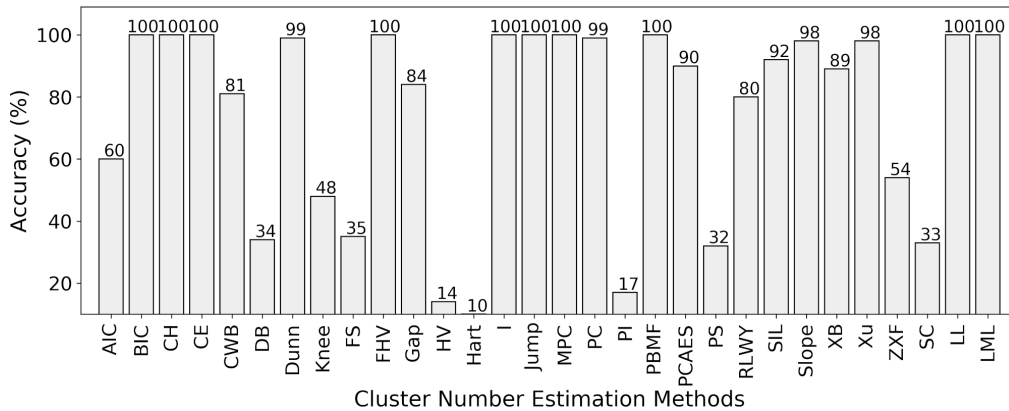


Figure 2.9: Accuracy of estimating the number of clusters on datasets satisfying all constraints.

methods when the clusters are well-separated, over increasing number of dimensions as well as increasing number of clusters. The results are also summarized in Table 2.3, where we observe that both LL and LML are among the methods that have obtained the lowest average rank, with comparable performances to the top nineteen methods and statistically significant performances in comparison to the rest of the methods. LL and LML also have the additional advantage of having very low $O(k^2)$ computation complexity. With increase in the size of the datasets, the running time for LL and LML is always drastically low compared to the rest of the top performing cluster number estimation methods. Table 2.4 shows the execution times of the top performing methods on increasing dataset sizes. We observe significantly lower execution times for LL, followed quite closely by LML, making them the best choice when working on large datasets.

Given the impressive results of LL and LML under ideal conditions, the next question that occurs naturally is how the performance drops when each of the ideal constraints are violated.

2.5.3 Violation of Constraint 1

In this experiment we first investigate the performance under soft violation of Constraint 1, by generating datasets with clusters having either 100 or 125 points, while keeping constant all

2. Fast Automatic Estimation of the Number of Clusters

Table 2.3: The summary of the performances of the cluster number estimation methods *when all constraints are satisfied*.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	7.67	1.00	1.00	1.00	4.44	17.76	1.45	13.68
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	1.92E-06	1.0000	1.0000	1.0000	0.9524	3.06E-12	0.9999	9.25E-05
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	1.92E-06	1.0000	1.0000	1.0000	0.9525	3.06E-12	0.9999	9.25E-05
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	22.01	1.00	14.57	22.10	18.23	1.00	1.00	1.00
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	7.65E-45	1.0000	7.68E-14	3.79E-09	0.5108	1.0000	1.0000	1.0000
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	7.65E-45	1.0000	7.68E-14	3.79E-09	0.5108	1.0000	1.0000	1.0000
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	1.00	26.11	1.00	5.18	22.32	5.40	6.56	1.67
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	1.0000	5.05E-47	1.0000	0.4416	1.14E-53	0.2739	2.13E-06	0.9999
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LML)	1.0000	5.05E-47	1.0000	0.4416	1.14E-53	0.2739	2.13E-06	0.9999
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	3.74	1.53	13.25	20.77	1.00	1.00		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	0.3957	0.9999	1.42E-04	2.01E-48	-	1.0000		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	0.3957	1.0000	1.42E-04	2.01E-48	1.0000	-		

Table 2.4: The Execution Times for the Top Performing Methods from Section 2.5.2. The Execution Times are Averaged over 30 runs of each Method on Datasets Generated with Increasing Sizes, Containing 5 Equal-sized Well-separated Spherical Clusters.

Methods	$n = 2^8$	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$
BIC	0.0037	0.0079	0.0163	0.0451	0.1359
CH	0.0033	0.0045	0.0115	0.0251	0.0875
CE	0.0034	0.0077	0.0266	0.1085	0.4605
FHV	0.0014	0.0035	0.0083	0.0404	0.1958
I	0.0026	0.0073	0.0133	0.0552	0.1944
MPC	0.0011	0.0026	0.0064	0.0354	0.1319
PBMF	0.0070	0.0187	0.0593	0.2648	0.9540
LL	0.0005	0.0007	0.0009	0.0018	0.0022
LML	0.0005	0.0008	0.0010	0.0022	0.0025

other experimental conditions. 50 groups of data are generated, each containing 25 datasets. The generated data is of 2, 10, 20, 35, and 50 dimensions, in each dimension 2, 10, 20, 35, and 50 clusters are generated. From the results in Figure 2.10 we observe similar performances for most methods. LL and LML show resilience to slight imbalances in the number of points in the clusters, performing as well as the other top-performing methods. Table 2.5 also summarizes the results, from which we observe both LL and LML are among the methods that have obtained the lowest rank, and show comparable performance with the top nineteen scoring methods.

In the next experiment, we strongly violate Constraint 1, by generating clusters with

2. Fast Automatic Estimation of the Number of Clusters

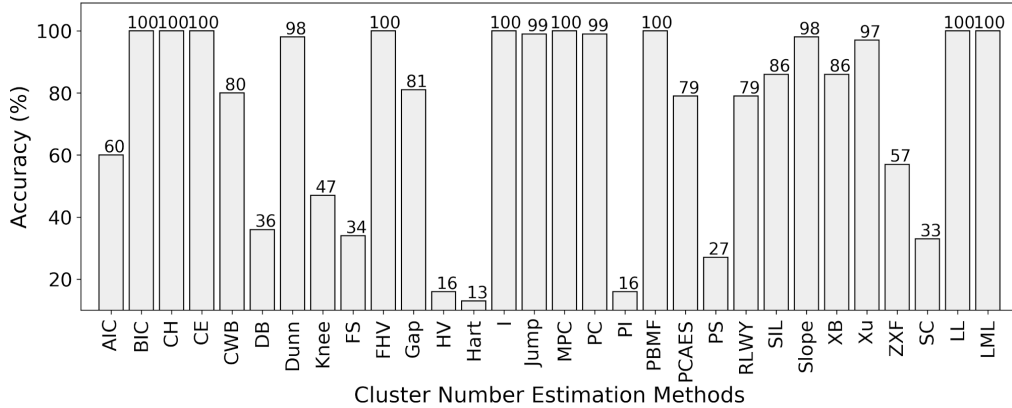


Figure 2.10: Accuracy of estimating the number of clusters on datasets with clusters having slight difference in the number of points.

Table 2.5: The summary of the performances of the cluster number estimation methods *when clusters have slight differences in the number of points*.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	5.99	1.00	1.00	1.00	4.85	17.09	1.19	13.95
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	2.33E-04	1.0000	1.0000	1.0000	0.9432	6.76E-12	0.9999	5.54E-05
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	2.33E-04	1.0000	1.0000	1.0000	0.9432	6.76E-12	0.9999	5.54E-05
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	21.76	1.00	12.59	22.45	18.56	1.00	1.00	1.00
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	3.49E-44	1.0000	1.44E-09	3.36E-10	0.5254	1.0000	1.0000	1.0000
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	3.49E-44	1.0000	1.44E-09	3.36E-10	0.5254	1.0000	1.0000	1.0000
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	1.00	26.24	1.00	5.71	22.72	4.31	6.59	1.16
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	1.0000	1.13E-46	1.0000	0.3390	1.04E-53	0.5987	2.26E-06	0.9999
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LML)	1.0000	1.13E-46	1.0000	0.3390	1.04E-53	0.5987	2.26E-06	0.9999
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	3.67	1.49	13.16	20.18	1.00	1.00		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	0.2619	0.9999	1.41E-04	6.84E-39	-	1.0000		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	0.2619	0.9999	1.41E-04	6.84E-39	1.0000	-		

100, 150, 200 or 250 points while keeping all other experiment conditions the same. Similar to Section 2.5.2 we generate 50 groups of data, covering increasing number of dimensions (2, 10, 20, 35, and 50) and increasing number of clusters (2, 10, 20, 35, and 50). We obtain the results shown in Figure 2.11. We observe promising performances similar to the results of the previous experiment, with only slight changes in the performances for some methods. This indicates that most methods are able to identify the correct number of clusters due to the general high accuracy of k -Means in clustering datasets with well-separated clusters. The results are also summarized in Table 2.6, where we observe that LL and LML are among

2. Fast Automatic Estimation of the Number of Clusters

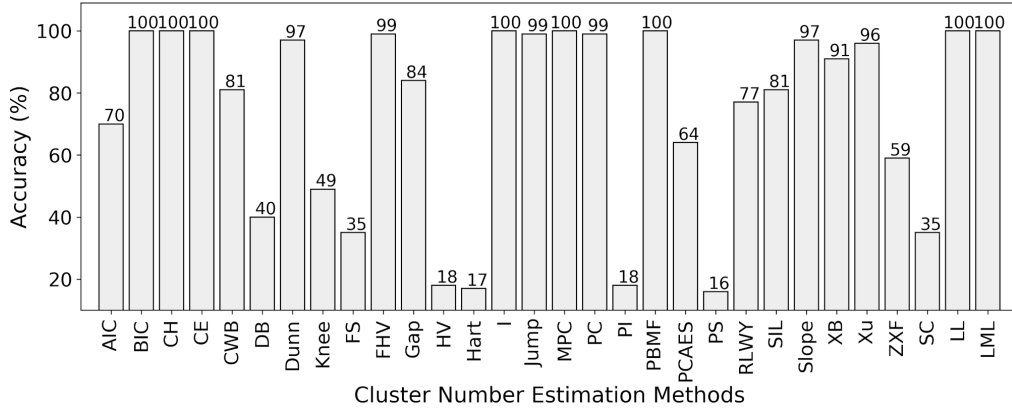


Figure 2.11: Accuracy of estimating the number of clusters on datasets with clusters having high difference in the number of points.

the lowest rank achieving methods, and have comparable performance with the top nineteen scoring methods.

Table 2.6: The summary of the performances of the cluster number estimation methods *when clusters have high differences in the number of points.*

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	5.96	1.00	1.00	1.00	4.65	15.88	1.15	13.26
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	1.18E-04	1.0000	1.0000	1.0000	0.9126	1.20E-12	0.9999	7.77E-05
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	1.18E-04	1.0000	1.0000	1.0000	0.9126	1.20E-12	0.9999	7.77E-05
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	21.35	1.00	12.97	23.13	18.61	1.00	1.00	1.00
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	4.56E-42	1.0000	1.57E-10	1.78E-13	0.3837	1.0000	1.0000	1.0000
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	4.56E-42	1.0000	1.57E-10	1.78E-13	0.3837	1.0000	1.0000	1.0000
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	1.00	26.29	1.00	6.17	23.13	5.67	6.53	1.64
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	1.0000	1.08E-49	1.0000	0.3297	2.69E-54	0.6096	3.12E-06	0.9999
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LML)	1.0000	1.08E-49	1.0000	0.3297	2.69E-54	0.6096	3.12E-06	0.9999
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	3.28	1.81	12.59	20.06	1.00	1.00		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	0.4759	0.9936	5.59E-05	2.03E-33	-	1.0000		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	0.4759	0.9936	5.59E-05	2.03E-33	1.0000	-		

2.5.4 Violation of Constraint 2

In the next experiment we violate constraint 2 by randomly generating clusters with the standard deviation set to 2, 3, or 4 times the minimum standard deviation (which is set to 1). We generate 50 groups of data in a manner similar to Section 2.5.2, covering increasing

2. Fast Automatic Estimation of the Number of Clusters

number of dimensions (2, 10, 20, 35, and 50) and increasing number of clusters (2, 10, 20, 35, and 50). The results in Figure 2.12 show that LL, CE, FHV, I and MPC performed the best. The performance of the previously top performing CH dropped slightly (-7%) in the presence of high variation in the dispersion of the clusters. We observe a rather high drop in performance for BIC. Here we observe an interesting contrast in the behaviour of LL and LML. Since LML identifies equal sized clusters, it tends to choose a higher number of clusters when neighbouring clusters have different spreads. This behaviour is beneficial in accurately selecting the number of clusters for a dataset such as Iris, but on synthetic datasets with a large disparity in the spread of nearby clusters, LML tends to break the larger clusters into smaller equal-sized clusters, as shown in Figure 2.13. Thus when we do not know much about the cluster structures present in a data, the difference in the estimated number of clusters by LL and LML can help inform us about the nature of the clusters identified by k -Means. The overall results are also summarized in Table 2.7, where LL obtains the lowest average rank with performance comparable to the top eighteen methods.

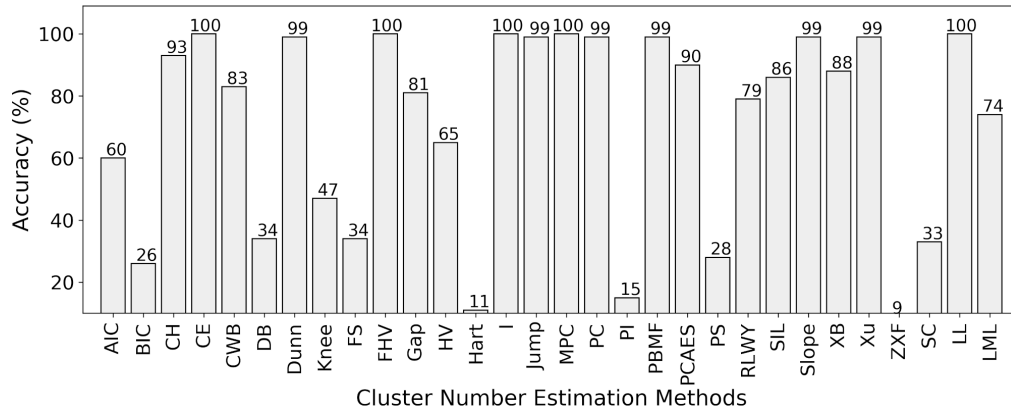


Figure 2.12: Accuracy of estimating the number of clusters on datasets where clusters have different spread of points.

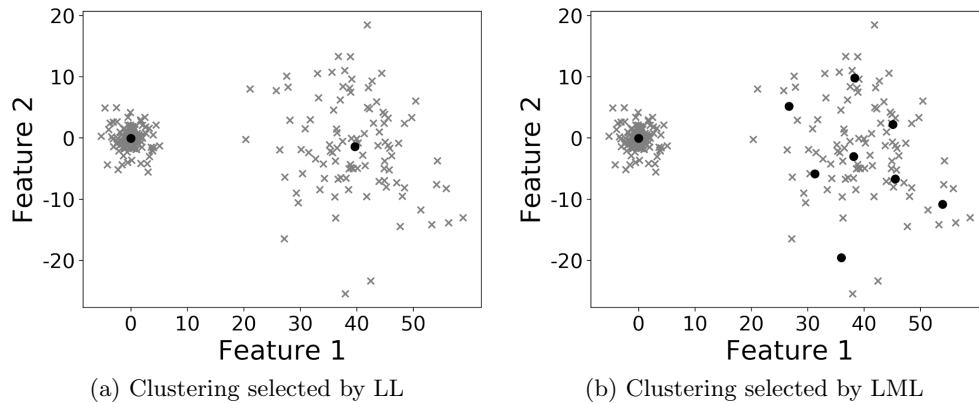


Figure 2.13: For neighbouring clusters with high disparity in spread, LL tends to identify well-separated clusters, whereas LML identifies clusters of similar size.

Table 2.7: The summary of the performances of the cluster number estimation methods *when clusters have differences in cluster variances*.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	8.43	15.95	2.10	1.00	3.41	17.61	1.32	13.29
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	4.57E-08	0.2284	0.8972	1.0000	0.9606	9.99E-11	0.9977	9.61E-05
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	3.13E-03	0.6729	0.2950	9.22E-02	5.95E-02	4.26E-14	8.82E-02	2.00E-08
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	21.65	1.00	12.62	5.97	17.45	1.00	1.00	1.00
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	2.03E-44	1.0000	1.62E-09	4.78E-02	0.5486	1.0000	1.0000	1.0000
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	1.37E-45	9.22E-02	1.06E-16	8.45E-03	4.69E-02	9.22E-02	9.22E-02	9.22E-02
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	1.00	26.43	1.00	4.28	21.71	5.54	6.57	1.15
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	1.0000	4.38E-61	1.0000	0.5205	1.43E-51	0.4594	2.14E-06	0.9999
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LML)	9.22E-02	6.60E-48	9.22E-02	1.96E-03	4.94E-45	1.72E-01	2.98E-07	9.24E-02
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	4.76	1.30	23.59	20.68	1.00	10.26		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	3.76E-01	0.9997	1.15E-09	8.32E-50	-	9.22E-02		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	6.54E-01	8.99E-02	3.03E-04	1.19E-50	9.22E-02	-		

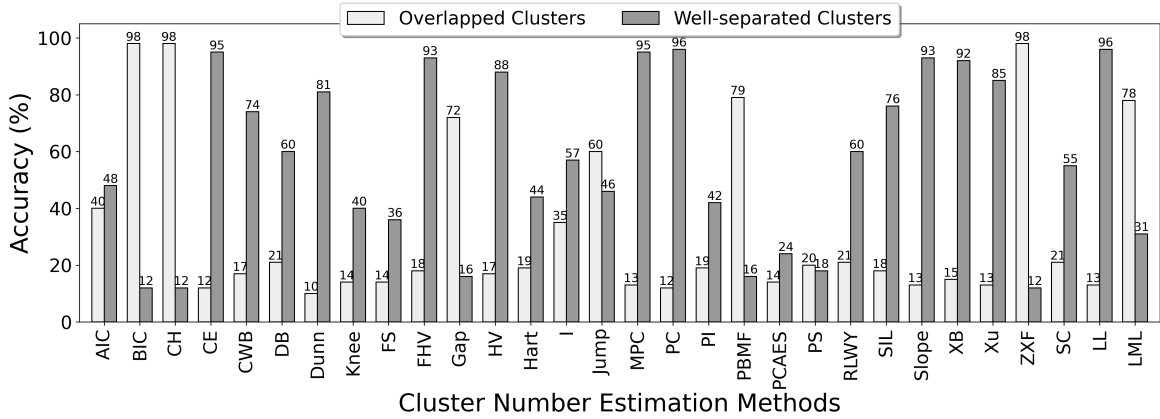


Figure 2.14: Accuracy of estimating the number of clusters on datasets with clusters having slight overlap.

2.5.5 Violation of Constraint 3

We first investigate the performances under soft violation of Constraint 3, by generating data with clusters having low overlap. This is ensured by generating clusters with a 50% probability of its center lying within a minimum distance between 3 and 5 units to the center of another cluster (the variance of all clusters is 1 unit). We generate 100 groups of data, where each group contains five two-dimensional datasets. Each dataset has 5, 10, 20, 35, and 50 number

2. Fast Automatic Estimation of the Number of Clusters

of clusters. Generating two-dimensional data makes it easier to manually verify how many of the clusters are overlapped, and how many are well-separated. From the results shown in Figure 2.14, we observe that BIC, CH and ZXF achieve the highest accuracy in identifying the number of overlapped clusters. Although ZXF and BIC perform well here, in previous experiments we observed lower performances for ZXF when the number of dimensions was greater than two, and for BIC when there were large disparities in cluster variances. CH is perhaps the most reliable in identifying the clusters when they are slightly overlapped. When looking at the number of well-separated clusters, we observe the previously top performing LL, CE, FHV, MPC, and PC perform the best here as well. Even in the presence of overlap, these methods can reliably identify the number of well-separated clusters. We observe that by dividing clusters into equal sizes, LML can provide a decent estimate of the number of overlapped clusters. The results of the experiments are also summarized in Tables 2.8 and 2.9. In Table 2.8 when overlapped clusters are to be identified, BIC, CH and Xu have obtained the lowest average rank, followed by LML which shows statistically significant performance to them. In Table 2.9 when well-separated clusters are to be identified, PC followed by MPC obtain the lowest average rank, followed by LL which has statistically comparable performance to both the top scoring methods.

Table 2.8: Performances of the cluster number estimation methods *when clusters have low degrees of overlaps*, measured in terms of identifying the number of overlapped clusters.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	5.85	1.13	1.13	12.75	17.02	15.09	16.79	21.80
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	1.06E-06	1.54E-12	1.54E-12	0.7495	8.50E-03	6.86E-12	0.7593	1.17E-16
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LML)	3.09E-09	2.60E-02	2.60E-02	1.71E-14	6.66E-14	8.06E-17	1.58E-14	7.76E-38
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	23.13	10.25	10.08	11.58	21.74	7.67	4.77	11.68
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	2.20E-16	5.61E-01	2.02E-02	3.84E-01	7.28E-06	9.66E-03	2.97E-07	7.38E-01
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	8.37E-49	1.44E-11	4.58E-07	6.88E-12	1.09E-23	1.21E-03	1.96E-01	4.87E-13
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	12.38	19.72	3.45	25.66	16.68	13.08	13.24	12.38
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	7.49E-01	2.01E-17	1.26E-10	1.94E-36	2.17E-21	9.48E-04	9.03E-04	7.54E-01
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0
p-val (LML)	5.73E-14	2.38E-28	9.97E-01	4.30E-138	1.44E-27	1.04E-04	5.28E-05	6.63E-13
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	11.13	15.31	1.13	15.38	12.55	2.06		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_1		
p-val (LL)	7.56E-01	7.81E-01	1.54E-12	4.71E-17	-	7.45E-12		
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-		
p-val (LML)	8.31E-11	1.58E-14	2.60E-02	3.85E-21	7.45E-12	-		

Next, we investigate the performance under strong violation of Constraint 3. Clusters are generated with 50% probability of their centers having a minimum distance between 2 and 3 units to the center of another cluster (the variance of all clusters is fixed at 1). Similar to the previous experiment, we generate 100 groups of two-dimensional datasets containing 5, 10, 20, 35, and 50 clusters. The results in Figure 2.15 suggest that only CH (with a 75% accuracy) and ZXF (with an accuracy of 86%) can occasionally provide correct estimates for

2. Fast Automatic Estimation of the Number of Clusters

Table 2.9: Performances of the cluster number estimation methods *when clusters have low degrees of overlaps*, measured in terms of identifying the number of well-separated clusters.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	15.73	22.00	22.00	5.41	8.54	9.98	7.42	14.55
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LL)	3.37E-50	5.13E-65	5.13E-65	1.73E-01	7.48E-09	2.82E-12	4.14E-05	2.51E-19
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LML)	5.71E-22	1.97E-01	1.97E-01	1.68E-63	3.35E-17	5.21E-10	4.72E-24	2.47E-02
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	15.98	4.32	21.48	4.42	17.22	11.51	14.84	1.89
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0
p-val (LL)	1.54E-20	1.80E-02	1.08E-64	2.35E-01	1.49E-23	3.56E-12	2.55E-13	1.38E-01
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LML)	1.24E-01	7.09E-52	1.17E-01	1.81E-64	1.57E-02	9.51E-10	1.42E-02	3.00E-68
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	1.73	13.24	20.30	20.71	19.32	14.48	8.45	4.62
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	3.64E-01	6.35E-17	3.75E-59	1.18E-32	5.39E-67	4.22E-06	2.36E-01	5.79E-01
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LML)	5.52E-70	1.27E-02	4.78E-01	6.24E-01	6.96E-03	2.03E-11	2.52E-33	1.28E-55
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	4.62	5.75	22.00	12.11	3.25	19.44		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_1		
p-val (LL)	1.41E-01	2.02E-03	5.13E-65	3.30E-16	-	2.34E-41		
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-		
p-val (LML)	8.17E-43	4.16E-35	1.97E-01	1.42E-06	2.34E-41	-		

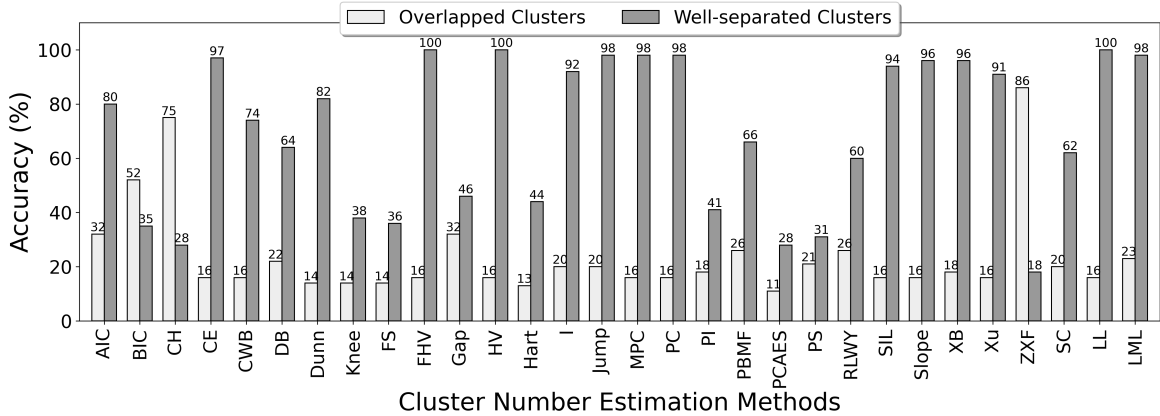


Figure 2.15: Accuracy of estimating the number of clusters on datasets with clusters having high overlap.

the number of overlapped clusters. We observe that LL, FHV, and HV perform the best at identifying the number of well-separated clusters, closely followed by LML, Jump, MPC, and PC. Since highly overlapped clusters form dense well-separated clusters, both LL and LML provide more accurate estimates. These results are also summarized in Tables 2.10 and 2.11. In Table 2.10 where overlapped clusters were to be identified, we observe that ZXF obtain the lowest average rank, followed by CH and BIC, which were also previously the top performing methods in identifying overlapped clusters when constraint 3 was weakly violated. In Table 2.11 where well-separated clusters were to be identified, our proposed LL is observed

2. Fast Automatic Estimation of the Number of Clusters

to have obtained the lowest average rank, and LML is among the top seven methods that have statistically comparable performances.

Table 2.10: Performances of the cluster number estimation methods *when clusters have high degrees of overlaps*, measured in terms of identifying the number of overlapped clusters.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	5.10	2.71	1.89	9.27	15.42	15.43	14.07	20.99
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LL)	2.44E-04	5.54E-07	5.70E-09	9.99E-01	1.71E-02	1.53E-08	6.46E-01	6.60E-23
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1
p-val (LML)	3.15E-03	2.76E-06	1.62E-08	9.35E-01	1.90E-02	1.50E-07	3.64E-01	3.91E-23
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	22.73	8.49	7.43	9.03	22.54	7.01	8.47	9.10
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	2.41E-25	9.97E-01	6.55E-04	9.57E-01	1.61E-05	3.81E-01	8.76E-01	9.88E-01
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	7.19E-27	9.45E-01	2.38E-05	9.83E-01	1.76E-06	4.00E-01	9.85E-01	9.62E-01
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	9.31	19.33	8.23	26.08	16.68	9.65	9.77	10.91
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	9.98E-01	2.28E-23	2.73E-02	1.90E-51	3.09E-25	2.34E-03	4.64E-01	9.86E-01
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	9.43E-01	2.27E-22	5.64E-02	8.79E-66	1.55E-23	1.15E-03	7.31E-01	9.49E-01
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	8.58	12.66	16.00	1.09	9.17	7.40		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	9.13E-01	8.69E-01	4.83E-11	1.08E-12	-	9.48E-01		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	8.75E-01	7.97E-01	5.02E-10	2.39E-11	9.48E-01	-		

Table 2.11: Performances of the cluster number estimation methods *when clusters have high degrees of overlaps*, measured in terms of identifying the number of well-separated clusters.

Method	AIC	BIC	CH	CE	CWB	DB	Dunn	Knee
Avg. Ranks	18.04	20.96	21.98	4.58	9.63	11.60	7.49	17.30
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LL)	6.73E-154	8.56E-91	2.15E-84	1.34E-01	2.11E-19	4.70E-16	4.13E-10	8.94E-43
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-val (LML)	1.18E-54	1.62E-61	1.65E-58	2.78E-01	3.54E-10	2.05E-12	2.55E-04	2.49E-30
Method	FS	FHV	GAP	HVZ	Hart	I	Jump	MPC
Avg. Ranks	19.15	2.11	21.84	1.61	20.20	8.85	3.17	2.72
Hyp. Test (LL)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	9.44E-53	2.91E-01	2.47E-62	8.43E-02	1.70E-56	2.57E-21	4.04E-01	1.07E-01
Hyp. Test (LML)	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	9.60E-36	4.76E-01	1.32E-47	1.16E-01	1.10E-24	3.80E-06	4.73E-01	2.49E-01
Method	PC	PI	PBMF	PCAES	PS	RLWY	Rez	SIL
Avg. Ranks	3.64	15.72	14.52	24.09	23.52	14.35	5.03	3.63
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LL)	8.19E-02	1.96E-33	2.11E-54	1.75E-87	2.78E-176	2.57E-21	6.32E-01	8.91E-01
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0
p-val (LML)	1.08E-01	2.84E-25	5.06E-25	1.27E-59	3.04E-103	1.62E-09	6.20E-01	8.33E-01
Methods	Slope	XB	Xu	ZXF	LL	LML		
Avg. Ranks	3.75	5.55	25.07	12.44	1.33	2.86		
Hyp. Test (LL)	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0		
p-val (LL)	9.33E-02	3.15E-74	3.00E-98	5.29E-18	-	5.45E-01		
Hyp. Test (LML)	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-		
p-val (LML)	2.20E-01	1.14E-01	1.39E-85	3.22E-16	7.32E-02	-		

Table 2.12: Specifications of the Real Datasets.

Name	Size of dataset
Data Banknote Authentication	(1372,4)
Echocardiogram	(106,9)
Iris	(150,4)
Seeds	(210,7)
Sonar Mines vs. Rocks	(208,60)
Wine	(178,13)
Colon Cancer	(62,2000)
Prostate Cancer	(102,6033)

2.5.6 Real-world data

The identification of the number of clusters from real-world datasets is indeed a challenge. Clustering methods should identify groups that are sufficiently dissimilar. Identifying clusters that are well-separated ensures that the identified clusters are sufficiently dissimilar. Therefore in this section we investigate the performance of LL and LML on identifying the number of clusters in real-world datasets. The real datasets considered are shown in Table 2.12. The first six datasets are from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017). The last two are high-dimension cancer datasets present at www.stat.cmu.edu/~jiashun/Research/software/GenomicsData (last accessed July 12, 2021). Since we consider the k -Means algorithm which identifies clusters of the same size, the data is preprocessed to remove classes that contain very low number of points in comparison to the other classes. Every real dataset is normalized, by mean-centering each feature which is then divided by the difference in the maximum and minimum value of the feature.

Table 2.13: The Average Rank of Estimating \hat{k} from Real Datasets.

Name	Average Rank	Name	Average Rank
AIC	18.000	MPC	2.750
BIC	18.000	PC	1.000
CH	2.875	PI	13.625
CE	1.000	PBMF	6.125
CWB	8.375	PCAES	8.375
DB	18.000	PS	4.125
Dunn	15.500	RLWY	2.875
Knee	4.250	SIL	18.000
FS	4.125	Slope	4.375
FHV	5.375	XB	2.625
Gap	5.500	Xu	8.375
HV	18.000	ZXF	18.000
Hart	6.625	SC	12.875
I	2.875	LL	1.000
Jump	4.375	LML	1.000

Due to the higher difficulty in estimating the number of clusters on real datasets, performing within $[k^* - 1, k^* + 1]$ is considered a success. Each method is assigned a rank based

on their performance, where the most successful methods are assigned rank 1, and then the next best methods are assigned rank 2, and so on. The average rank for each method is reported in Table 2.13. From the results we observe LL, LML, CE, and PC perform the best, closely followed by XB, MPC, CH, I and RLWY. Thus, we observe that LL and LML achieved competent performance on real datasets as well. The full results for each method on all the real datasets are present in Section A.3 of Appendix A.

2.6 Discussion

In this chapter, we propose two methods: LL and LML to estimate the number of clusters from a dataset. Both methods use only the minimum distance between cluster centers to estimate the number of clusters. We compared different cluster number estimation methods that are applicable for the k -Means clustering algorithm to investigate which method performs the best. Among intra-cluster variance based cluster estimation methods, we note that the popular knee-point method does not perform well in general, which is justifiable since there exists very little theoretical reasons for it to work. Some intra-cluster variance based methods such as BIC and CH are observed to work well, especially to identify clusters that are overlapped. One of the major advantages of the proposed inter-cluster based LL and LML methods is that can estimate the number of clusters at a significantly lower cost of computation, and we have also provided a detailed explanation in Section 2.3 on how they operate. From extensive experiments, we observe that LL and LML can estimate the number of clusters from different perspectives, giving a more informed view of the cluster structures present in a given data. LL performs at par with the best cluster number estimation methods in accurately identifying the number of well-separated clusters. LML performs equally well in ideal situations, and in more ambiguous situations it can identify equal-sized clusters. The difference in results between LL and LML can inform a user of the underlying cluster structures present in a dataset. The performances of LL and LML are consistent over a number of challenging situations (such as slight or high differences in the number of points in different clusters, different spreads of clusters, and slight or high overlap between clusters). We also observe that LL and LML are robust to the increase in the number of clusters (up to 50 clusters) as well as the number of dimensions (up to 50), while drastically outperforming all other methods in terms of computation time required due to its $O(k^2)$ time complexity, where generally $k \ll n$. We conclude that to identify clusters that are well-separated, the minimum distance between clusters provides enough information to estimate the number of clusters. This lets us make the general recommendation of using LL or LML in applications to identify clusters from large datasets. In such situations LL and LML can efficiently estimate the number of clusters by considering only the minimum distance between centers, while not needing to consult the dataset.

Chapter 3

On the Unification of k -Harmonic Means and Fuzzy c -Means Clustering Problems under Kernelization

Summary

We present a common algorithm for the kernel k -harmonic means (KKHM) and the kernel fuzzy c -means (KFCM) clustering problems. We incorporate kernel functions in a generalized fuzzy c -means cost function, forming the cost function of a kernelized general fuzzy c -means (KGFCM) problem, and design an algorithm to locally minimize this cost function. The KGFCM cost function has two parameters: the exponent p of the Euclidean distance, and the fuzzy weighting exponent m . By setting proper values for p and m in our algorithm, one can execute the KKHM or the KFCM algorithm. Using the algorithm for KKHM, we compare its clustering performance with the popular kernel k -means and KFCM algorithms. Experiments performed on real-world and synthetic datasets show the superior clustering capabilities of KKHM. We also show that KKHM retains the advantages of the original KHM algorithm, resulting in better clustering performance when a high number of clusters are present.

3.1 Introduction

One of the factors that affects center-based clustering is the distance metric used, as was discussed in Section 1.2 of Chapter 1. k -Means clustering (MacQueen, 1967; Jain, 2010) uses the squared Euclidean distance to determine which data instances are close to the cluster centers that are being estimated. The squared Euclidean distance is a well-defined metric, however there exists other distance or similarity metrics that are *parameterized*, which allows them to be tuned for the specific requirements of a dataset under consideration. In this chapter we study the possibility of using a more flexible distance metric for a center-based clustering problem called k -Harmonic Means (KHM), which was introduced by Zhang et al. (1999). In KHM the cost function is defined as the harmonic means of the squared Euclidean distance between data instances and cluster centers. The resulting algorithm was empirically

more robust towards the position of initial centers compared to Lloyd’s algorithm for k -means (Lloyd, 1982). They also showed that KHM led to superior clustering when the number of clusters was large. Zhang (Zhang, 2000, 2001) constructed a general KHM cost function using the harmonic means of the p -th power of the Euclidean distance. For $p > 2$, higher weights are assigned to data instances farther away from a center, leading to lower chances of a single high density cluster trapping multiple centers. Zhi and Fan (2010) provided theoretical justifications for this behaviour, showing that the derivative of an M-estimate of the KHM cost function became unbounded for values of $p > 1$, indicating higher influence on centers by instances lying farther away.

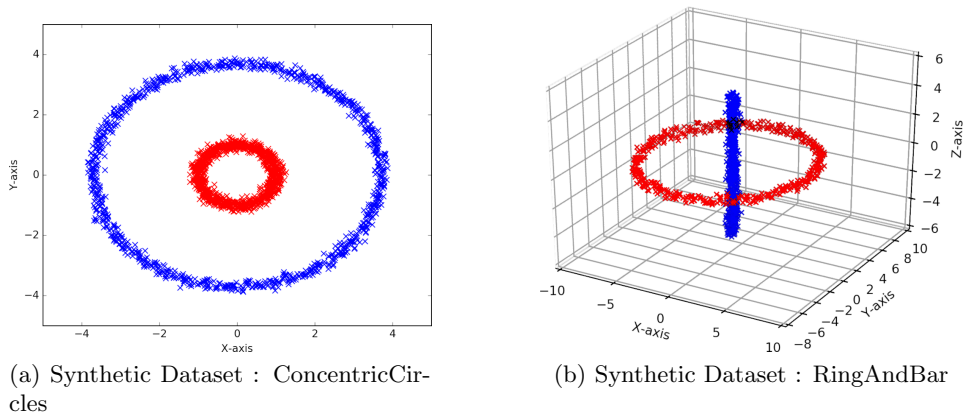


Figure 3.1: Synthetic Non-linearly Separable Datasets.

One disadvantage of k -means is that it fails to find clusters that are not linearly separable, such as the ones shown in Figure 3.1, since it divides the space of data instances into Voronoi regions around each cluster center. A variant of k -means is fuzzy c -means (FCM) (Dunn, 1973; Bezdek, 1981) where clusters are defined as fuzzy sets. Data instances can have memberships to multiple clusters, leading to the detection of overlapping clusters. However, FCM also fails to find non-linearly separable clusters. One way of overcoming this limitation is to use kernel functions (Müller et al., 2001; Vapnik, 1998). The idea behind kernel functions is that non-linearly separable clusters can become linearly separable when mapped to a higher dimensional space. By computing the similarities between data instances mapped to a higher dimensional Hilbert space, and using an inverse mapping to the original space, one can obtain measures of similarity between data instances. This process is computationally intensive, but can be computed feasibly by the use of kernel functions. There are numerous kernel functions available; Table 3.1 shows three widely used ones. Improved cluster detection was reported when kernel functions were used in k -means and FCM (Zhang and Rudnicky, 2002; Dhillon et al., 2004; Graves and Pedrycz, 2010; Zhang and Chen, 2004; Yang and Tsai, 2008; Ferreira and de A.T. de Carvalho, 2014). Applying kernel functions in other clustering methods (MacDonald and Fyfe, 2000; Kim et al., 2005; Cleuziou and Moreno, 2015; Ferreira et al., 2016) led to successful detection of non-linearly separable clusters as well.

Li et al. (2007) investigated the performance of a modified kernel KHM targeted towards the specific objective of image segmentation. However, their work was based on a KHM cost function with $p = 2$. Hamerly and Elkan (2002) showed that the general KHM cost function using the p -th power of the Euclidean distance results in superior clustering compared to

Table 3.1: Popular Kernel Functions.

Name	Kernel Function
Gaussian kernel	$K(a, b) = \exp(-\ a - b\ ^2 / 2\sigma^2)$
Polynomial kernel	$K(a, b) = (a \cdot b + c)^d$
Sigmoid kernel	$K(a, b) = \tanh(c(a \cdot b) + d)$

Note: a, b are data instances. σ, c, d are kernel function parameters.

k -means and FCM. Given the empirical evidence and the supporting theory on the benefits of using the p -th power of the Euclidean distance, we believe that a study on the performance of a kernelized KHM can be worthwhile.

In this chapter, we introduce a new cost function for the kernel k -harmonic means (KKHM) problem, by incorporating kernel functions in a generalized FCM (GFCM) cost function (Zhi and Fan, 2010), based on a general c -Means model (Yu and Yang, 2005). We show the kernelized cost function is equivalent to the KKHM cost function, and propose a common algorithm for kernel fuzzy c -means (KFCM) and KKHM. The rest of the chapter is organized as follows: Section 3.2 provides a brief overview of KHM and two popular kernel center-based clustering methods, kernel k -means and KFCM. Next, we discuss the GFCM cost function. In Section 3.3 we extend the GFCM cost function to form the cost function of the kernelized GFCM, from which we form the cost function for KKHM. We then propose a common algorithm for KKHM and KFCM. In Section 3.4, we report the experimental results comparing the performances of KKHM, kernel k -means and KFCM with appropriate discussions.

3.2 Background

In this section we discuss KHM along with two popular kernel clustering algorithms, k -means and FCM. Next, we discuss the generalized FCM cost function.

3.2.1 The k -harmonic means problem

Let $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^T$ be the dataset of vectors of numerical features, where $\mathbf{x}_i \in \mathbb{R}^d$. We want to find k clusters represented by k centers $V = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)^T$, $v_i \in \mathbb{R}^d$. The KHM problem is to minimize the cost function

$$J_{KHM_p}(V) = \sum_{i=1}^n \frac{k}{\sum_{l=1}^k \frac{1}{\|\mathbf{x}_i - \mathbf{v}_l\|^p}}. \quad (3.1)$$

Generally $p \geq 2$. For larger values of p , the cost function is more sensitive to outliers and noise. An algorithm for KHM can be designed from an update rule for the centers that can be formed from the derivative of the cost function. The algorithm starts from an initial set of centers and applies the update rule iteratively till the change in centers is less than a predetermined threshold.

3.2.2 The kernel k -means problem

Kernel functions are integrated into the cost function of the k -means problem, by *kernelization in the feature space* (Schölkopf et al., 1998). Cluster centers are considered to exist in a higher dimensional Hilbert space referred to as the *feature space*. A non-linear function ϕ maps the data instances to the feature space, where the Euclidean distances from the centers are computed. The sum of such distances for all centers gives the cost function,

$$J_{KKM}(V) = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\phi(\mathbf{x}_i) - \mathbf{v}_j\|^2. \quad (3.2)$$

Here the k clusters are denoted by C_1, C_2, \dots, C_k . Derivatives of the cost function lead to the eqn. for the centers,

$$\mathbf{v}_j = \frac{\sum_{\mathbf{x}_i \in C_j} \phi(\mathbf{x}_i)}{|C_j|}. \quad (3.3)$$

The Euclidean distance $d_{i,j}^\phi$ from $\phi(\mathbf{x}_i)$ to \mathbf{v}_j can be computed by substituting the expression for \mathbf{v}_j from eqn. (3.3) to eqn. (3.2).

$$d_{i,j}^\phi = \left\| \phi(\mathbf{x}_i) - \frac{\sum_{\mathbf{x}_s \in C_j} \phi(\mathbf{x}_s)}{|C_j|} \right\|^2 \quad (3.4)$$

$$= K(\mathbf{x}_i, \mathbf{x}_i) - 2 \frac{\sum_{\mathbf{x}_s \in C_j} K(\mathbf{x}_i, \mathbf{x}_s)}{|C_j|} + \frac{\sum_{\mathbf{x}_s, \mathbf{x}_t \in C_j} K(\mathbf{x}_s, \mathbf{x}_t)}{|C_j|^2}. \quad (3.5)$$

Using the *kernel trick*, every dot product $\phi(\mathbf{x}_i)^T \cdot \phi(\mathbf{x}_i')$ in the expansion of eqn. (3.4) is replaced by a kernel function in eqn. (3.5). Kernel functions such as the ones in Table 3.1 can be used here. An algorithm for kernel k -means is formed by starting with an initial cluster assignment of all data instances, then iterate over applying eqn. (3.5) with a specific predetermined kernel function to assign data instances to the closest cluster.

3.2.3 The kernel fuzzy c -means problem

There are two ways of incorporating kernel functions in FCM. One is by *kernelization in the feature space*, which is similar to how k -means was kernelized in eqn. (3.2), leading to the following cost function,

$$J_{KFCM_m}^{(1)}(V) = \sum_{i=1}^N \sum_{j=1}^c \mu_{ij}^m \|\phi(\mathbf{x}_i) - \mathbf{v}_j\|^2. \quad (3.6)$$

Here the number of clusters is c , and the matrix $U = (\mu_{ij})_{n \times c}$ is the matrix of fuzzy membership values, denoting the fuzzy membership of data point \mathbf{x}_i in cluster C_j , and $\sum_{j=1}^c \mu_{ij} = 1, \forall i$.

The second method is *kernelization of the metric*, where the center is in the input space. ϕ maps both the centers and the data instances to the feature space, where the distances are computed. This method of kernelization is relevant to our work, as it makes it possible to

design the algorithm for KKH. The cost function to be minimized is given by,

$$J_{KFCM_m}^{(2)}(V) = \sum_{i=1}^N \sum_{j=1}^c \mu_{ij}^m \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^2 . \quad (3.7)$$

If Gaussian kernel functions are considered, the derivative of the kernel function is $\nabla_b K(a, b) = \frac{(a-b)}{\sigma^2} K(a, b)$. Also note that $K(a, a) = 1$. The squared expression in eqn. (3.7) can be expanded and after using the kernel trick, from its derivative the update expressions for μ_{ij} and \mathbf{v}_j can be derived.

$$\mu_{ij} = \frac{(1 - K(\mathbf{x}_i, \mathbf{v}_j))^{-\frac{1}{m-1}}}{\sum_{l=1}^c (1 - K(\mathbf{x}_i, \mathbf{v}_l))^{-\frac{1}{m-1}}} , \quad (3.8a)$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^n \mu_{ij}^m K(\mathbf{x}_i, \mathbf{v}_j) \mathbf{x}_i}{\sum_{i=1}^n \mu_{ij}^m K(\mathbf{x}_i, \mathbf{v}_j)} . \quad (3.8b)$$

From an initial set of centers, the above two update expressions can be executed in every iteration until the algorithm converges.

3.2.4 The generalized fuzzy c-means problem

The cost function for a generalized fuzzy c -means (GFCM) problem (Zhi and Fan, 2010) is,

$$J_{GFCM_{m,p}}(U, V) = \sum_{i=1}^n \sum_{j=1}^c \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^p , \quad \sum_{j=1}^c \mu_{ij} = 1 . \quad (3.9)$$

Setting $p = 2$ forms the cost function for FCM. Zhi and Fan (2010) showed that reformulating the cost function and setting $m = 2$ gives the cost function for KHM. They equated the derivatives of the Lagrangian of the cost function with respect to μ_{ij} to zero, and derived an expressions for μ_{ij} .

$$\mu_{ij} = \frac{1}{\sum_{l=1}^c \left(\frac{\|\mathbf{x}_i - \mathbf{v}_j\|^p}{\|\mathbf{x}_i - \mathbf{v}_l\|^p} \right)^{\frac{1}{m-1}}} . \quad (3.10)$$

By substituting the expression of μ_{ij} from eqn. (3.10) into eqn. (3.9), they reformulate the cost function as J_{RGFCM} .

$$J_{RGFCM_{m,p}} = \sum_{i=1}^n \left(\sum_{j=1}^c (\|\mathbf{x}_i - \mathbf{v}_j\|^p)^{\frac{1}{1-m}} \right)^{\frac{1}{1-m}} . \quad (3.11)$$

Setting $m = 2$, they derive the cost function of KHM.

$$J_{RGFCM_{2,p}} = \sum_{i=1}^n \frac{1}{\sum_{j=1}^c \frac{1}{\|\mathbf{x}_i - \mathbf{v}_j\|^p}} = J_{KHM_p} . \quad (3.12)$$

In the next section, this cost function J_{GFCM} is extended to form the cost function for KKHM.

3.3 Kernel k-harmonic means (KKHM)

In this section we discuss our original contributions: a cost function for KKHM, and a common algorithm for KKHM and KFCM.

3.3.1 A cost function for KKHM

Directly incorporating a kernel function in eqn. (3.1) makes it difficult to find an update expression for the centers. To overcome this difficulty, we incorporate kernel functions to J_{GFCM} to get the cost function of a kernelized general FCM (KGFCM) problem,

$$J_{KGFCM_{m,p}} = \sum_{i=1}^n \sum_{j=1}^c \mu_{ij}^m \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p , \quad \sum_{j=1}^c \mu_{ij} = 1 . \quad (3.13)$$

Note that with $p = 2$ we get $J_{KFCM_m}^{(2)}$. We reformulate (Zhi and Fan, 2010) this cost function to form the cost function of the KKHM clustering problem. Equating the derivatives of the Lagrangian of the cost function with respect to μ_{ij} to zero, we get an update expression for μ_{ij} ,

$$\mu_{ij} = \left(\sum_{l=1}^c \left(\frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_l)\|^p}{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p} \right)^{\frac{1}{m-1}} \right)^{-1} . \quad (3.14)$$

Substituting this expression for μ_{ij} in eqn. (3.13), we get,

$$\begin{aligned}
 J_{KGF_{CM_{m,p}}} &= \sum_{i=1}^n \sum_{j=1}^c \frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p}{\left[\sum_{l=1}^c \left(\frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_l)\|^p}{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_1)\|^p} \right)^{\frac{1}{m-1}} \right]^m} \\
 &= \sum_{i=1}^n \sum_{j=1}^c \frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^{p-\frac{pm}{m-1}}}{\left[\sum_{l=1}^c \frac{1}{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_l)\|^{\frac{p}{m-1}}} \right]^m} \\
 &= \sum_{i=1}^n \frac{\sum_{j=1}^c \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^{\frac{p}{1-m}}}{\left[\sum_{l=1}^c \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_l)\|^{\frac{p}{1-m}} \right]^m} \tag{3.15} \\
 &= \sum_{i=1}^n \left[\left(\sum_{j=1}^c \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p \right)^{\frac{1}{1-m}} \right]^{1-m} \\
 &= J_{RKGFCM_{m,p}} .
 \end{aligned}$$

Setting $m = 2$, we get the cost function for KKH_M,

$$J_{RKGFCM_{2,p}} = \sum_{i=1}^n \frac{1}{\sum_{j=1}^c \frac{1}{\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p}} = J_{KKHM_p} . \tag{3.16}$$

3.3.2 A common algorithm for KKH_M and KFCM

From $J_{KGF_{CM_{m,p}}}$ in eqn. (3.13), we form the Lagrangian,

$$\mathcal{L} = \sum_{i=1}^n \sum_{j=1}^c \mu_{ij}^m \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p - \sum_{i=1}^n \lambda_i \left(\sum_{j=1}^c \mu_{ij} - 1 \right) . \tag{3.17}$$

Using the kernel trick, $\|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^p = [K(\mathbf{x}_i, \mathbf{x}_i) - 2K(\mathbf{x}_i, \mathbf{v}_j) + K(\mathbf{v}_j, \mathbf{v}_j)]^{p/2}$. Setting the derivatives of the Lagrangian to zero, and using Gaussian kernels, we can find the update expressions for the membership values μ_{ij} and centers \mathbf{v}_j .

$$\mu_{ij} = \frac{\left[[1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{\frac{p}{2}} \right]^{-\frac{1}{m-1}}}{\sum_{l=1}^c \left[[1 - K(\mathbf{x}_i, \mathbf{v}_l)]^{\frac{p}{2}} \right]^{-\frac{1}{m-1}}} , \tag{3.18a}$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^n \mu_{ij}^m [1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{\frac{p-2}{2}} K(\mathbf{x}_i, \mathbf{v}_j) \mathbf{x}_i}{\sum_{i=1}^n \mu_{ij}^m [1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{\frac{p-2}{2}} K(\mathbf{x}_i, \mathbf{v}_j)} . \tag{3.18b}$$

By setting $m = 2$ in eqns. (3.18), for $p \geq 2$, the algorithm executes update conditions for

KKHM,

$$\mu_{ij} = \frac{[1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{-\frac{p}{2}}}{\sum_{l=1}^c [1 - K(\mathbf{x}_i, \mathbf{v}_l)]^{-\frac{p}{2}}}, \quad (3.19a)$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^n \mu_{ij}^2 [1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{\frac{p-2}{2}} K(\mathbf{x}_i, \mathbf{v}_j) \mathbf{x}_i}{\sum_{i=1}^n \mu_{ij}^2 [1 - K(\mathbf{x}_i, \mathbf{v}_j)]^{\frac{p-2}{2}} K(\mathbf{x}_i, \mathbf{v}_j)}. \quad (3.19b)$$

Also note that setting $p = 2$ in eqns. (3.18), we get eqns. (3.8) which are the update conditions for KFCM.

The following algorithm¹ serves as a common algorithm for both KFCM and KKHM. With $m = 2$, the algorithm executes KKHM, and with $p = 2$, the algorithm executes KFCM. The algorithm randomly initializes centers, then iteratively computes the kernel function for all instances and centers, and updates the centers and membership values using eqns. (3.18). Computing the kernel function for n instances and c centers takes $O(ncd)$ time. Updating all μ_{ij} takes $O(nc^2d)$ time, and updating all \mathbf{v}_j takes $O(ncd)$ time, leading to an overall $O(nc^2d)$ time for the KGFCM algorithm. This is at par with the FCM algorithm, with computation of kernel functions taking some additional time, but requiring the same $O(nc)$ space. By contrast, the kernel k -means algorithm requires initial computation of kernel functions between every two instances, requiring $O(n^2cd)$ time, and additional $O(n^2)$ space. The proof of convergence follows directly from the proof for FCM (Gröll and Jäkel, 2005). The key differences are the introduction of $\phi(\cdot)$ and the use of the p^{th} power of the Euclidean distance, which do not affect the proof of convergence.

Algorithm 1 The KGFCM clustering algorithm

Note: Set $m = 2$ for KKHM; set $p = 2$ for KFCM.

Input: n instances $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$; number of clusters c ; parameters m, p ; tolerance ϵ .

Output: c centers $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c$.

- 1: Randomly initialize c centers $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c$.
 - 2: **repeat**
 - 3: Compute $K(\mathbf{x}_i, \mathbf{v}_j), \forall \mathbf{x}_i, \mathbf{v}_j$.
 - 4: Update $\mu_{ij}, \forall i, j$ using eqn. (3.18a).
 - 5: Update $\mathbf{v}_j, \forall j$ using eqn. (3.18b).
 - 6: **until** $\sum_j \|\mathbf{v}_j^{\text{new}} - \mathbf{v}_j^{\text{prev}}\|^2 < \epsilon$.
-

3.4 Experiment and Results

In this section, the clustering performance of KKHM is compared with kernel k -means (KKM) and KFCM. We use Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) to quantify

¹Source codes are available at: https://github.com/Avisek20/kernelized_general_fuzzy_c_means

clustering performances. Section 3.4.1 outlines the datasets used for the comparison. Section 3.4.2 describes the experimental procedure on real and clustering datasets, and Section 3.4.3 describes the experiments performed on synthetic datasets with large number of clusters.

3.4.1 Datasets

21 datasets from the UCI Machine Learning repository (Dheeru and Karra Taniskidou, 2017), and the Joensuu datasets repository (at <https://cs.joensuu.fi/sipu/datasets/>, last accessed November 12, 2021) are used to compare the methods, shown in Table 3.2. 4 synthetic datasets are used: ConcentricCircles, RingAndBar, BIRCHlike, and BIRCHlike:diffDensities. The datasets ConcentricCircles and RingAndBar contain non-linearly separable clusters, shown in Figure 3.1. BIRCHlike contains 49 clusters, each containing 200 data instances. BIRCHlike:diffDensities also contains 49 clusters, where clusters are either sparse with around 50 data instances, or dense with around 200 data instances. Datasets like BIRCHlike are generally difficult to cluster accurately with center-based clustering methods, and BIRCHlike:diffDensities presents an even more difficult scenario where cluster centers will tend to be drawn towards denser clusters.

Table 3.2: Summary of Datasets.

Dataset	Dataset Size	Number of clusters
Aggregation	(788,2)	7
Banknote Authentication	(1372,4)	2
Breast Cancer Wisconsin	(683,9)	2
Compound	(399,2)	6
D31	(3100,2)	31
Ecoli	(336,7)	8
Flame	(240,2)	2
Iris	(150,4)	3
Jain Shape Sets	(373,2)	2
Leaf	(340,14)	30
Occupancy	(8143,5)	2
Optical Handwritten Recognition	(1797,64)	10
Parkinsons	(195,22)	2
Pathbased	(300,2)	3
R15	(600,2)	15
Seeds	(210,7)	3
Statlog Image Segmentation	(2310,19)	7
Statlog Landsat Satellite	(4435,36)	7
Image Segmentation	(210,19)	7
Vertebral Column	(310,6)	3
Wine	(178,13)	3
ConcentricCircles	(2000,3)	2
RingAndBar	(1000,3)	2
BIRCHlike	(10282,2)	49
BIRCHlike:diffDensities	(6543,2)	49

3.4.2 Comparison of kernel clustering algorithms

To compare the clustering algorithms, we executed 20 runs of KKH, KKM, and KFCM. Each run had a maximum of 200 iterations, with $0.01 \leq \sigma \leq 50$ in increments of 0.01. For KKH we considered $2 < p \leq 10$ with increments of 1, and for KFCM we considered $2 \leq m \leq 10$ with increments of 1. We recorded the maximum ARI for each method. Table 3.3 shows the results of our study. Gaussian kernels were used for the experiment.

From the results, we observe that KKH generally performs better than KFCM or KKM, performing equally well or slightly worse on only a few. We observe that KKH often requires higher values of p to achieve the best clustering results. On non-linearly separated clusters such as ConcentricCircles and RingAndBar, using kernel functions resulted in proper separation by all three clustering methods. On the synthetic datasets BIRCHlike and BIRCHlike:diffDensities, KKH performed much better than KFCM and KKM, which can be observed in Figs. 3.2 and 3.3.

Table 3.3: Comparison of Performances using ARI.

Dataset	KKH	p	KFCM	m	KKM
Aggregation	0.950	2	0.790	7	0.754
Banknote Authentication	0.702	2	0.309	2	0.859
Breast Cancer Wisconsin	0.908	4	0.885	2	0.858
Compound	0.845	6	0.742	2	0.854
D31	0.954	2	0.783	2	0.907
Ecoli	0.745	2	0.745	2	0.500
Flame	0.950	3	0.950	2	0.900
Iris	0.960	2	0.960	2	0.960
Jain Shape Sets	0.790	4	0.726	2	0.758
Leaf	0.405	8	0.345	4	0.365
Occupancy	0.785	2	0.785	2	0.672
Optical Handwritten Recognition	0.706	8	0.291	2	0.803
Parkinsons	0.499	7	0.420	5	0.424
Pathbased	0.930	4	0.699	7	0.726
R15	0.992	2	0.885	2	0.883
Seeds	0.811	8	0.406	7	0.439
Statlog Image Segmentation	0.549	8	0.428	2	0.517
Statlog Landsat Satellite	0.562	2	0.562	3	0.579
Image Segmentation	0.676	5	0.484	2	0.642
Vertebral Column	0.429	6	0.410	8	0.380
Wine	0.917	8	0.881	2	0.659
ConcentricCircles	1.0	2	1.0	2	1.0
RingAndBar	1.0	2	1.0	2	1.0
BIRCHlike	0.813	4	0.683	2	0.784
BIRCHlike:diffDensities	0.904	4	0.665	2	0.869

The method of KKH has two parameters, the parameter p on the distance metric, and the parameter σ for the Gaussian kernel metric. The performance of KKH depends on the setting of suitable values to these two parameters, however this is not trivial. As shown in Figure 3.4, for the dataset D31 the ARI obtained by KKH across a range of values for σ for different values of p does not show any discernible relationship between p and σ which can be utilized to suggest suitable values for p and σ . The variations in ARI changes across the datasets in consideration as well, for which as an example we consider a different dataset of

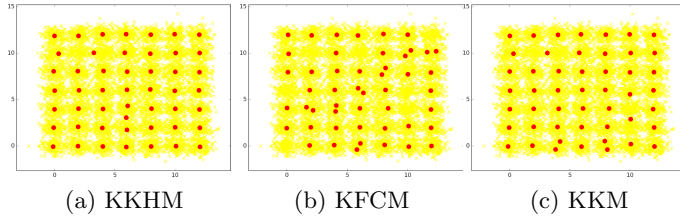


Figure 3.2: Comparison on synthetic dataset BIRCHlike.

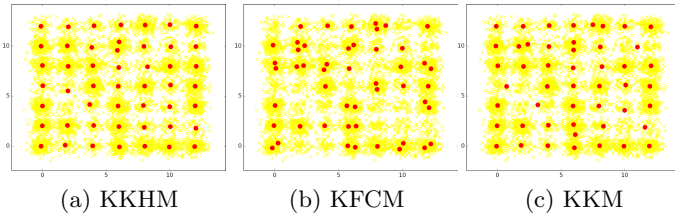


Figure 3.3: Comparison on synthetic dataset BIRCHlike:diffDensities.

Statlog Landsat Satellite in Figure 3.5. As can be seen from the figure, for different values of p , the ARI changes across the range of values of σ in different ways. Therefore we conclude that a grid search is recommended to obtain suitable values for p and σ , which for the same reasons is also common practice to set the parameters for the state-of-the-art multiple kernel clustering methods Liu et al. (2017b); Yao et al. (2020).

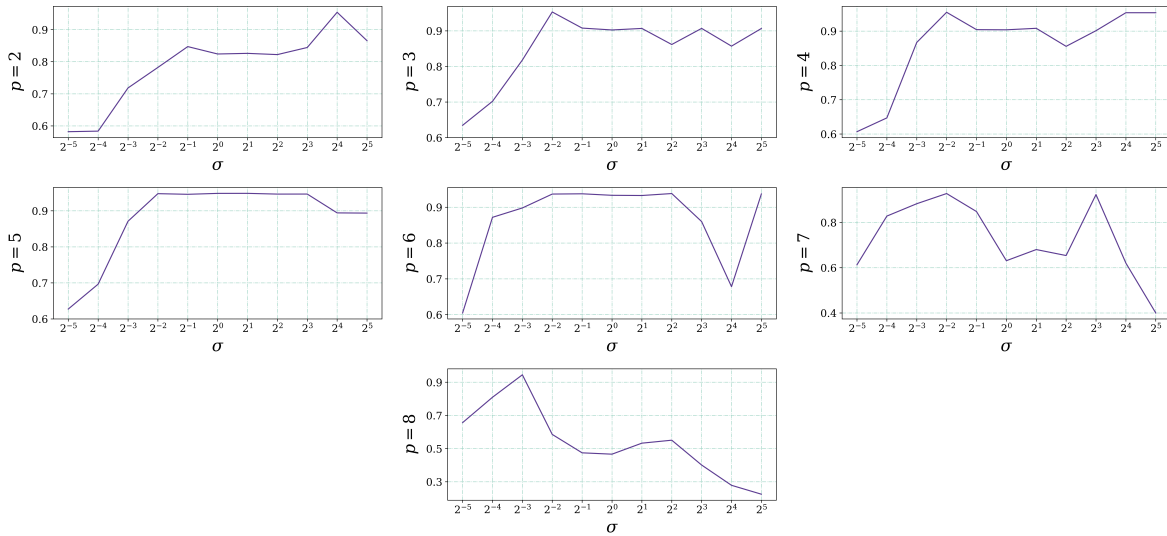


Figure 3.4: For the dataset D31, the variations in ARI achieved by KKHM for σ ranging from 2^{-5} to 2^5 is shown for different values of p ranging from $p = 2$ to $p = 8$. A relationship established between p and σ could potentially lead to more informed approaches to select suitable values for p and σ . However as can be seen from the plots, there is no discernible relationship between p and σ , thus leading to a recommendation of using a grid search to obtain suitable values for p and σ .

3. On the Unification of KHM and FCM under Kernelization

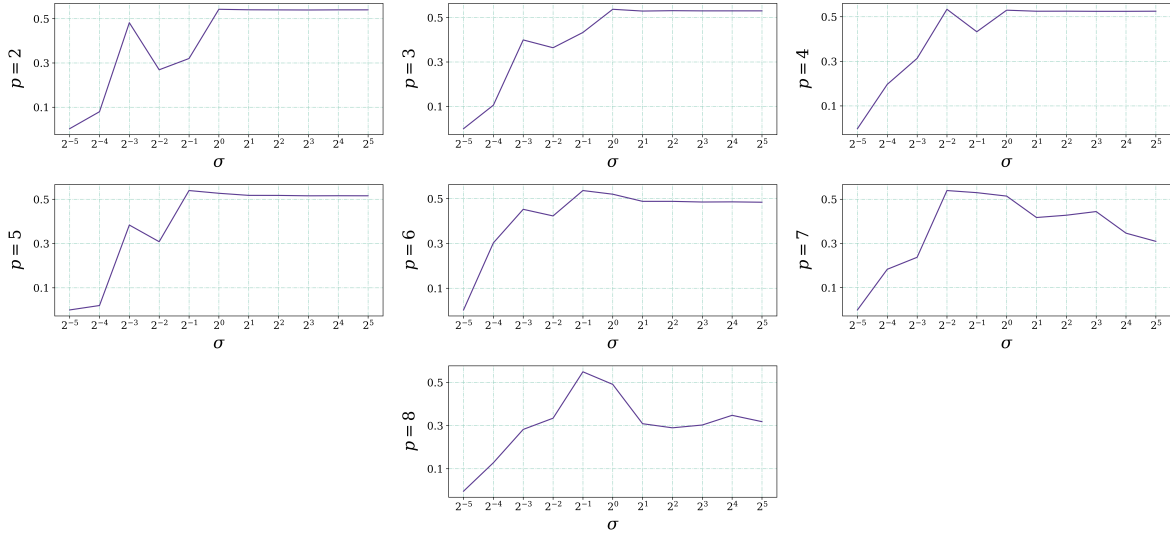


Figure 3.5: For the dataset Statlog Landsat Satellite, the variations in ARI achieved by KKHMM are shown for σ ranging from 2^{-5} to 2^5 and for values of p between $p = 2$ and $p = 8$. Similar to our observations for the dataset D31, there is no discernible relationship between p and σ . For each p the maximum ARI is observed for values of σ between 2^{-2} and 2^1 , whereas the maximum ARI was often outside this interval for the dataset D31. This leads us to conclude that different datasets can have dissimilar optimal values of p and σ . Therefore a grid search is recommended to obtain suitable values for p and σ .

3.4.3 Performance on large number of clusters

We designed an experiment to test the performance of KKHMM when a large number of clusters are present. 49 two-dimensional clusters were randomly generated, where each cluster could be either dense or sparse with equal probability. Dense clusters contained between 200 to 250 instances, decided randomly. Similarly sparse clusters contained between 40 and 50 instances, decided randomly (e.g., Figure 3.6.).

We randomly generated 1000 such datasets, and on all datasets we ran KKHMM, KFCM, KKM and BIRCH (Zhang et al., 1996). Birch generally performs well when a large number of clusters are present. For KKHMM, p was varied from 3 to 8 and the best ARI was noted. σ for Gaussian kernels was set to the variance of the data. All algorithms were run 5 times, for a maximum of 500 iterations, with a tolerance threshold of 10^{-6} . BIRCH was run with branching factor set to 50, and threshold set to 0.5.

From the results of the experiment we observed that KKHMM gave the best ARI for 999 of the 1000 datasets. KKM gave the best ARI for 1 dataset, and KFCM and BIRCH failed to give the maximum ARI for any. Figure 3.7 shows a box plot of the ARI achieved over the 1000 generated datasets. KKHMM gives the best average ARI with the lowest standard deviation ($0.8676 (\pm 0.0108)$). In comparison, KKM shows a slight edge in performance ($0.8133 (\pm 0.0241)$) compared to BIRCH ($0.7926 (\pm 0.0265)$), and KFCM resulted in the lowest ARI values ($0.6708 (\pm 0.0402)$).

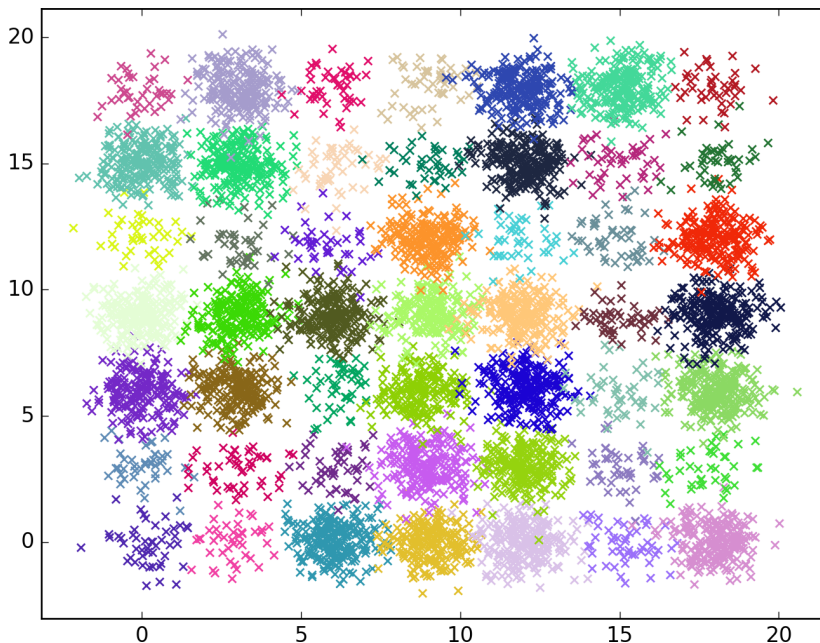


Figure 3.6: A randomly generated dataset containing sparse and dense clusters.

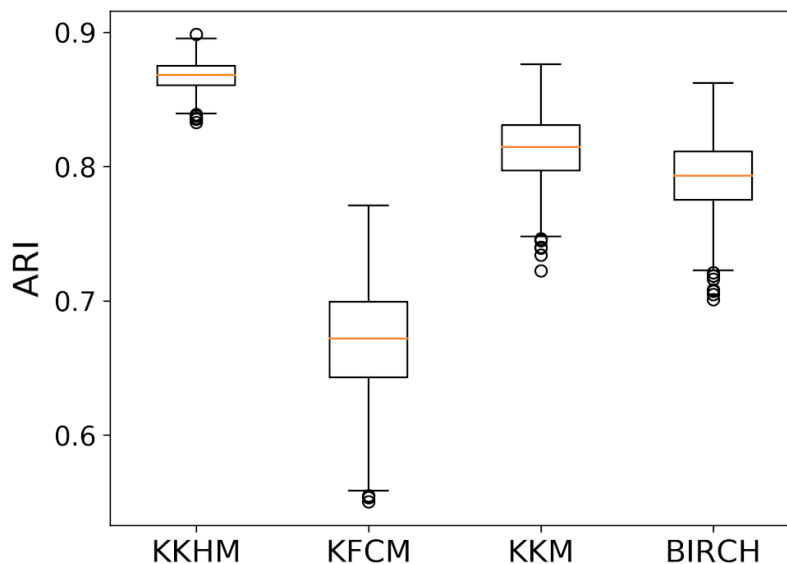


Figure 3.7: Difference in ARI when running KKHM, KFCM, KKM and BIRCH. Plot depicts ARI for 200 randomly chosen datasets for clarity.

3.5 Discussion

In this chapter, we introduced a common algorithm for the KKHM and KFCM problems. Our experiments involving real-world and synthetic datasets show that the KKHM algorithm outperforms the popular KFCM and kernel k -means algorithms over several datasets. KKHM performs significantly better for datasets with a high number of clusters, outper-

forming BIRCH as well. This property is thus retained from KHM. The common algorithm for KKHM and KFCM has lower space and time complexity and higher cluster detection capabilities than kernel k -means, making it a better alternative.

KHM has generally been investigated less often in areas of applications of data clustering such as image segmentation, graph clustering, gene sequence analysis, etc. Given the success of kernel functions for other clustering algorithms, further investigations on the KKHM algorithm in such areas will be worthwhile, along with research in the selection of parameter values for the kernel functions ([Sarkar and Pal, 2011](#); [Pal and Sarkar, 2014](#)).

Chapter 4

Improved Efficient Model Selection for Sparse Hard and Fuzzy Center-Based Clustering

Summary

The class of center-based clustering offers methods to efficiently identify clusters in datasets, making them applicable on larger datasets. While a dataset may contain several features, not all of them may be equally informative or helpful towards cluster detection. Therefore, sparse center-based clustering methods offer a way to select only those features that may be useful in identifying the clusters present in a dataset. However to automatically determine the degree to which features should be selected, these methods use the Permutation Method which involves generating and clustering multiple randomly permuted datasets, leading to much higher computation costs. In this chapter, we propose an improved approach towards model selection for sparse clustering by using expressions of Bayesian Information Criterion (BIC) derived for the center-based clustering methods of k-Means and Fuzzy c-Means. The derived expressions of BIC require significantly lower computation costs, yet allow us to compare and select a suitable clustering from sparse clusterings that have selected varying number of features. Experiments on several synthetic and real-world datasets show that using BIC for sparse clustering model selection leads to remarkable improvements in the identification of sparse clusterings for both Sparse k-Means and Sparse Fuzzy c-Means.

4.1 Introduction

In Section 1.2 of Chapter 1 we discussed one of the factors that affect center-based clustering to be the data features that are considered. For a specific problem it is common to measure and collect different features, motivated by the notion that more features can provide more information about the problem at hand. However, not all features may be important or equally helpful in the identification of clusters. Therefore different approaches have been proposed in order to handle the initial features provided to a clustering algorithm to help in accurately identifying the clusters present in the data. Among these approaches, feature weighting (Huang et al., 2005) in clustering methods assign higher weights to features deemed

more important (Wang et al., 2004; de Amorim, 2016; Wang et al., 2019). This idea was extended to the general notion of subspace clustering (Domeniconi et al., 2004; Vidal, 2011; Elhamifar and Vidal, 2013), where different clusters are identified in different subspaces of the original data space (Deng et al., 2016; Jia and Cheung, 2018; Abdolali and Rahmati, 2020; Deng et al., 2020). A particular subclass of feature weighted clustering problems is the class of sparse clustering, where some features can be set to zero if deemed irrelevant to the identification of clusters (Arias-Castro and Pu, 2017; Gaynor and Bair, 2017). A trivial example of such a feature is a feature whose value remains constant across all data instances. An important work in sparse clustering is the Sparse k -Means problem (Witten and Tibshirani, 2010), where the ℓ_1 penalty (Tibshirani, 1996) of the feature weight vector is constrained to a sparsity parameter s , which decides the degree of sparsity of the feature weight vector, i.e., how many features are set to zero. Important advantages of the Sparse k -Means problem formulation are the derivation of algorithms that simultaneously select appropriate feature weights and perform clustering, as well as the convergence guarantees that can be obtained for them. The advantages of this clustering framework further inspired the recent formulation of the Sparse Fuzzy c -Means algorithm (Qiu et al., 2015; Chang et al., 2017a).

The simultaneous feature selection and clustering of Sparse k -Means and Sparse Fuzzy c -Means requires specification of the upper bound to the ℓ_1 -norm of the feature weights, as it is non-trivial to automatically determine the optimal degree of sparsity during the optimization procedure. To consider a wide range of possible degrees of sparsity, Sparse k -Means and Sparse Fuzzy c -Means optimize clusterings at different candidate upper bounds to the ℓ_1 -norm and obtain a set of optimized candidate clusterings at each degree of sparsity. The clustering methods then use the Permutation Method to select from this set a final clustering at an appropriate degree of sparsity. The Permutation Method involves computing the GAP statistic (Tibshirani et al., 2001) for the optimal clusterings obtained at each candidate degree of sparsity, and the clustering which measures the largest GAP statistic is deemed the most appropriate. The primary drawback of the Permutation Method is its high computational cost, due to its necessity of generating and clustering multiple random permutations of the dataset at each candidate upper bound to the ℓ_1 -norm, in order to compute the GAP statistic for each candidate degree of sparsity.

Therefore we are motivated by the following challenges. First, we wish to obtain a model selection approach for sparse clustering that is more efficient than the Permutation Method, and is therefore more applicable to larger datasets. Second, during the model selection approach, we wish to investigate whether the sparse clusterings can all be compared in the original data space, instead of being compared in the individual feature spaces corresponding to the features selected by each of the sparse clusterings. Thus, in this chapter we make the following original contributions:

1. We derive expressions of the Bayesian Information Criteria (BIC) Schwarz (1978) for k -Means and Fuzzy c -Means to select an optimal sparse clustering from a set of clusterings optimized at different degrees of sparsity.
2. Whereas the existing approach of sparse model selection compares the models locally at each degree of sparsity, we propose a global comparison of clustering models in the original data space. Specifically, we investigate the capabilities of the derived expressions of BIC for k -Means and Fuzzy c -Means to compare clusterings at different levels

of sparsity in the original data space.

3. The computational costs of the derived expressions of BIC are drastically lower in comparison to the Permutation Method, making the proposed methods of Sparse k -Means using BIC (SKM+BIC) and Sparse Fuzzy c -Means using BIC (SFCM+BIC) better suited for the sparse clustering of larger datasets.

The rest of the chapter is organized in the following way. In Section 4.2 we discuss the Sparse k -Means and Sparse Fuzzy c -Means problems, and in detail discuss the Permutation Method. In Section 4.3 we discuss our proposed approach of efficient sparse clustering model selection using BIC and derive expressions of BIC to be used for Sparse k -Means and Sparse Fuzzy c -Means. In Section 4.4 we test for the effectiveness of global comparisons of optimal sparse clusterings in the original data space, as well as the overall sparse clustering performance of SKM+BIC in comparison to state-of-the-art hard sparse clustering methods, and likewise, we test the sparse clustering performance of SFCM+BIC in comparison to state-of-the-art fuzzy sparse clustering methods, on several real-world and synthetic benchmark datasets. The source codes for our proposed methods and data are available at https://github.com/Avishek20/sparse_clustering_BIC.

4.2 Related Work

In this section, we discuss the general sparse clustering framework, from which the problem objectives and methods of Sparse k -Means and Sparse Fuzzy c -Means are derived. This is followed by a discussion on how the Permutation Method is used to select a clustering corresponding to an appropriate degree of sparsity.

4.2.1 Sparse Clustering Framework and Methods

In [Witten and Tibshirani \(2010\)](#) a sparse clustering framework was proposed, which follows the general form for the optimization criteria:

$$\begin{aligned} & \max_{\Theta} \sum_{l=1}^p w_l f_l(X_{*,l}, \Theta), \\ & \text{subject to } \|\mathbf{w}\|_2^2 \leq 1, \|\mathbf{w}\|_1 \leq s, \forall j w_j \geq 0. \end{aligned} \quad (4.1)$$

Here $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is a collection of data instances, where $\mathbf{x}_i \in \mathbb{R}^p$, w_l is the weight assigned to feature l , $f_l(X_{*,l}, \Theta)$ is a function involving only feature l of the data $X_{*,l}$, and Θ is the set of cluster estimators. From this framework, the problems of Sparse k -Means was formulated as,

$$\begin{aligned} & \max_{C, V, w} \sum_{l=1}^p w_l \left\{ \sum_{i=1}^n (x_{il} - \bar{x}_l)^2 - \sum_{j=1}^k \sum_{x_i \in C_j} (x_{il} - v_{jl})^2 \right\}, \\ & \text{subject to } \|\mathbf{w}\|_2^2 \leq 1, \|\mathbf{w}\|_1 \leq s, \forall l w_l \geq 0. \end{aligned} \quad (4.2)$$

Here $V = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is the set of cluster centers to be estimated for the k clusters $C = \{C_1, \dots, C_k\}$. An Alternative Optimization (AO) algorithm can be derived to optimize the Sparse k -Means criterion, which alternately updates the cluster memberships C , the centers V , and the feature weights \mathbf{w} . The derived update expressions for C and V are,

$$C_j = \{\mathbf{x}_i \mid \|\mathbf{w}\mathbf{x}_i - \mathbf{w}\mathbf{v}_j\|^2 \leq \|\mathbf{w}\mathbf{x}_i - \mathbf{w}\mathbf{v}_t\|^2 \quad \forall t\}, \quad (4.3)$$

$$\mathbf{v}_j = \sum_{\mathbf{x}_i \in C_j} \frac{\mathbf{x}_i}{|C_j|}. \quad (4.4)$$

To derive an update rule for \mathbf{w} , the variables C and V are held constant, which reduces the objective (4.2) to,

$$\begin{aligned} & \max_{\mathbf{w}} \sum_{l=1}^p w_l (a_l + 2\zeta), \\ & \text{subject to } \|\mathbf{w}\|_2^2 \leq 1, \|\mathbf{w}\|_1 \leq s. \end{aligned}$$

Here $a_l = \sum_{i=1}^n (x_{il} - \bar{x}_l)^2 - \sum_{j=1}^k \sum_{x_i \in C_j} (x_{il} - v_{jl})^2$. It can be shown that this objective is optimized when $\|\mathbf{w}\|_2^2 = 1$ and $\|\mathbf{w}\|_1 = s$. The optimal feature weights are computed by searching for a scalar Δ :

$$\mathbf{w} = \frac{\|S(\mathbf{a}, \Delta)\|_1}{\|S(\mathbf{a}, \Delta)\|_2}. \quad (4.5)$$

S here is a soft-thresholding operator $S(\mathbf{a}, \Delta) = \text{sign}(\mathbf{a})(|\mathbf{a}| - \Delta)_+$. The ℓ_1 -norm of w lies between a maximum of $\|\mathbf{w}\|_1^{\max} = \frac{\|\mathbf{a}\|_1}{\|\mathbf{a}\|_2}$ and a minimum of $\|\mathbf{w}\|_1^{\min} = 1$. When $\Delta = 0$, $\|\mathbf{w}\|_1$ is set to $\|\mathbf{w}\|_1^{\max}$. When Δ is set to the second-largest component of \mathbf{a} , $\|\mathbf{w}\|_1$ is set to $\|\mathbf{w}\|_1^{\min}$. Therefore to set $\|\mathbf{w}\|_1$ to $s \in [\|\mathbf{w}\|_1^{\min}, \|\mathbf{w}\|_1^{\max}]$, we simply need to perform a dichotomy search for the corresponding value for Δ .

A similar formulation for the recent method of Sparse Fuzzy c -means (Qiu et al., 2015; Chang et al., 2017a) was derived from the sparse framework (4.1) as follows:

$$\begin{aligned} & \max_{U, V, \mathbf{w}} \sum_{l=1}^p w_l \left\{ \sum_{i=1}^n (x_{il} - \bar{x}_l)^2 - \sum_{j=1}^k \sum_{i=1}^n \mu_{ij}^m (x_{il} - v_{jl})^2 \right\}, \\ & \text{subject to } \sum_{j=1}^k \mu_{ij} = 1, 0 \leq \mu_{ij} \leq 1, \\ & \|\mathbf{w}\|_2^2 \leq 1, \|\mathbf{w}\|_1 \leq s, \forall l \quad w_l \geq 0. \end{aligned} \quad (4.6)$$

Here m is the degree of fuzzification, and $U = [\mu_{ij}]_{(n \times k)}$ is the matrix of fuzzy cluster memberships $\mu_{ij} \in [0, 1]$ for every data point i to each cluster j . The Sparse Fuzzy c -Means objective can also be optimized using an AO algorithm that alternately updates U , V , and \mathbf{w} . Update rules for U and V are derived from the Lagrangian of the cost, while holding \mathbf{w}

constant:

$$\mu_{ij} = \left\{ \frac{\sum_{l=1}^p w_l (x_{il} - v_{jl})^2}{\sum_{t=1}^k \sum_{l=1}^p w_l (x_{il} - v_{tl})^2} \right\}^{-\frac{1}{m-1}}, \quad (4.7)$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^n \mu_{ij} \mathbf{x}_i}{\sum_{i=1}^n \mu_{ij}}. \quad (4.8)$$

Update rules for \mathbf{w} are derived by holding U and V constant to reduce the objective (4.6) to,

$$\begin{aligned} & \max_{\mathbf{w}} \sum_{l=1}^p w_l b_l, \\ & \text{subject to } \|\mathbf{w}\|_2^2 \leq 1, \|\mathbf{w}\|_1 \leq s. \end{aligned}$$

Here $b_l = \{\sum_{i=1}^n (x_{il} - \bar{x}_l)^2 - \sum_{j=1}^k \sum_{i=1}^n \mu_{ij}^m (x_{il} - v_{jl})^2\}$. Similar to Sparse k -Means, it can be shown that this objective is optimized when $\|\mathbf{w}\|_2^2 = 1$ and $\|\mathbf{w}\|_1 = s$. The ℓ_1 norm of \mathbf{w} lies in $[\|\mathbf{w}\|_1^{\min}, \|\mathbf{w}\|_1^{\max}]$, where $\|\mathbf{w}\|_1^{\min} = 1$ and $\|\mathbf{w}\|_1^{\max} = \frac{\|\mathbf{b}\|_1}{\|\mathbf{b}\|_2}$. A dichotomy search for Δ can help set the required value of $\|\mathbf{w}\|_1$ to $s \in [\|\mathbf{w}\|_1^{\min}, \|\mathbf{w}\|_1^{\max}]$ using the equation:

$$\mathbf{w} = \frac{\|S(\mathbf{b}, \Delta)\|_1}{\|S(\mathbf{b}, \Delta)\|_2}. \quad (4.9)$$

4.2.2 Permutation Method to select the degree of sparsity

The sparse clustering methods require the specification of the upper bound s to the ℓ_1 -norm of feature weights \mathbf{w} . The selection of this upper bound is, however, non-trivial. In [Witten and Tibshirani \(2010\)](#) the Permutation Method was proposed to select a suitable upper bound, the steps of which are:

1. Select n_s candidate upper bounds $\{s_1, s_2, \dots, s_{n_s}\}$.
2. Create B permuted datasets from the original dataset, by selecting a random permutation of each feature.
3. For each candidate upper bound s_i , cluster the B permuted datasets along with the original dataset.
4. Select the clustering for the s_i that maximizes the Gap Statistic:

$$\begin{aligned} & \arg \max_{i \in \{1, \dots, n_s\}} GAP_{(i)}. \\ & \text{where, } GAP_{(s_i)} = \frac{1}{B} \sum_{b=1}^B \log(J^b) - \log(J^*) \end{aligned} \quad (4.10)$$

Here J^* is the cost of clustering the original data, and J^b is the cost of clustering the b -th permuted dataset.

The Permutation Method thus involves generating data with random permutations of every feature, preserving the feature-wise variance while losing the original cluster structures. The Gap Statistic selects the clustering of the original data which differs the most from the clustering of the permuted datasets. The Gap Statistic uses the logarithm of the clustering cost to measure this difference. The major drawback of this approach is its high computation complexity. While the original data is clustered by center-based clustering methods in $O(knp)$ time, the permutation technique yields an $O((B+1)n_s knp)$ running time. If $B = 10$ and $n_s = 10$ is used to achieve reasonable estimates, this leads to a 110 times increase in the computation time compared to the underlying non-sparse center-based clustering method. This makes the application of the permutation technique to large datasets impractical.

4.3 Proposed Model Selection for Sparse Clustering Models

In this section, we discuss in detail our proposed approach to select a clustering from the set of clusterings optimized at different degrees of sparsity. While the Permutation Method involves computing the GAP statistic for the clusterings at different candidate upper bound $\{s_1, \dots, s_{n_s}\}$, our proposed sparse clustering model selection approach involves comparing clustering models globally in the original data space using expressions of BIC that have drastically lower computational costs. To discuss our proposed approach, we first formalize the notion of comparing different clustering models under an information criterion involving the likelihood of observing the sample data at hand, given the model parameters. We then derive expressions for Bayesian Information Criteria (BIC) to be used in Sparse k -Means and Sparse Fuzzy c -Means to select an appropriate clustering from the set of clusterings optimized at each candidate upper bound $\{s_1, \dots, s_{n_s}\}$.

4.3.1 Bayesian Information Criterion for clustering model selection

We assume the dataset of n instances $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^p$, is sampled from a mixture of k possible component distributions C_1, \dots, C_k . The probability of sampling each data point is,

$$P(\mathbf{x}_i) = \sum_{j=1}^k P(\mathbf{x}_i | \mathbf{x}_i \in C_j) P(\mathbf{x}_i \in C_j),$$

such that the mixture proportions sum to one: $\sum_{j=1}^k P(\mathbf{x}_i \in C_j) = 1$. From the law of total probability, the probability of observing all n samples is,

$$L(X; \Theta) = \prod_{i=1}^n \sum_{j=1}^k P(\mathbf{x}_i | \mathbf{x}_i \in C_j) P(\mathbf{x}_i \in C_j),$$

where Θ contains all parameters of the mixture distribution. The function $L(\cdot)$ is called the likelihood function. Given different sets of possible mixture model parameters, the best fit model can be identified as the one that maximizes the likelihood function. For ease of

estimation, usually the log of the likelihood function is considered.

$$l(X; \Theta) = \sum_{i=1}^n \log \sum_{j=1}^k P(\mathbf{x}_i | \mathbf{x}_i \in C_j) P(\mathbf{x}_i \in C_j).$$

The Bayesian Information criterion is defined as,

$$BIC(\Theta) = l(X; \Theta) - \frac{|\Theta|}{2} \log |X|, \tag{4.11}$$

where $|\Theta|$ is the number of parameters of the mixture distribution, and $|X|$ is the sample size. When comparing mixture models with the same set of parameters that differ only in parameter values, the second term remains constant, and only the log likelihood function $l(X; \Theta)$ changes.

From eqn. (4.11), we observe that model selection using BIC will involve *global* comparisons of models on the same data X . We can compare this possible usage of BIC with the Permutation Method, where clusterings are evaluated using GAP *locally*, i.e., at every degree of sparsity. To illustrate this notion, let us consider Sparse k -Means running at n_s possible upper bounds, then the Permutation method performs the following evaluations of GAP locally at every degree of sparsity s_i , as described in eqn. (4.10):

$$GAP_{(s_i)}(\{C^*, V^*, \mathbf{w}^*\}, \{C^1, V^1, \mathbf{w}^1\}, \dots, \{C^B, V^B, \mathbf{w}^B\}).$$

In contrast to evaluating clusterings locally, we can also consider evaluating clusterings *globally* across different degrees of sparsity, and investigate any possible improvements in identifying clusterings. The challenge of directly using an expression of BIC is that for different degrees of sparsity the model as well as the data space changes as the feature weights \mathbf{w} change. As an illustration, let the optimal parameters returned by Sparse k -Means across a range of candidate upper bounds $1, \dots, n_s$ be $\Theta_{(1)}, \dots, \Theta_{(n_s)}$. The set of parameters include the optimal feature weights $\mathbf{w}_{(1)}, \dots, \mathbf{w}_{(n_s)}$. Then the calculation of BIC involves calculations of the following log-likelihoods, where both the model parameters as well as the data space changes:

$$l(\mathbf{w}_{(1)}^T X; \Theta_{(1)}), \dots, l(\mathbf{w}_{(n_s)}^T X; \Theta_{(n_s)}).$$

These log-likelihoods cannot be directly compared since both the models and the data space changes across different candidate upper bounds to the l_1 -norm. An alternate approach is therefore to globally compare BIC across different degrees of sparsity in the original data space and comparing the clustering results. These comparisons can be done using BIC expressions derived for k -Means and Fuzzy c -Means respectively. This is easy to do since the feature weights do not influence the update expressions of the cluster centers of sparse k -Means as shown in eqn. (4.4) and those of sparse Fuzzy c -Means as shown in eqn. (4.8). Therefore these cluster centers also exist in the original data space, and together with the cluster memberships, they can be used to evaluate BIC in the original data space.

Therefore we propose to compute BIC for k -Means across different degrees of sparsity in the original data space, using the optimal cluster centers and memberships obtained from

Sparse k -Means clustering:

$$BIC_{(1)}^{KM}(\{C_{(1)}, V_{(1)}\}), \dots, BIC_{n_s}^{KM}(\{C_{(n_s)}, V_{(n_s)}\}). \quad (4.12)$$

We select the appropriate degree of sparsity as that for which we obtain maximum BIC.

$$s := \arg \max_s BIC_{(s)}^{KM}. \quad (4.13)$$

The same argument holds for using BIC for Sparse Fuzzy c -Means as well. Algorithms 2 and 3 outline our proposed approaches to use BIC to perform sparse k -Means and sparse Fuzzy c -Means.

Algorithm 2 Sparse k -Means using BIC (SKM+BIC)

Input: n data instances $\mathbf{x}_1, \dots, \mathbf{x}_n$; number of clusters k ; n_s upper bounds to the ℓ_1 -norm of feature weights s_1, \dots, s_{n_s} .

Output: cluster centers $V = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, cluster memberships $C = \{C_1, \dots, C_k\}$, feature weights $\mathbf{w} = (w_1, \dots, w_p)$.

- 1: **for** $s = 1, \dots, s_{n_s}$ **do**
 - 2: 1. Perform Sparse k -Means AO to obtain optimal C, V and \mathbf{w} from eqns. (4.3), (4.4) and (4.5).
 - 3: 2. Compute $BIC_{(s)}^{KM}(\{C_{(s)}, V_{(s)}\})$ using Remark 4.1.
 - 4: **end for**
 - 5: Select $s := \arg \max_s BIC_{(s)}^{KM}$, and return the corresponding $V_{(s)}, C_{(s)}, \mathbf{w}_{(s)}$.
-

Algorithm 3 Sparse Fuzzy c -Means using BIC (SFCM+BIC)

Input: n data instances $\mathbf{x}_1, \dots, \mathbf{x}_n$; number of clusters k ; n_s upper bounds to the ℓ_1 -norm of feature weights s_1, \dots, s_{n_s} .

Output: cluster centers $V = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, fuzzy cluster memberships $U = [\mu_{ij}]_{(n \times k)}$, feature weights $\mathbf{w} = (w_1, \dots, w_p)$.

- 1: **for** $s = 1, \dots, s_{n_s}$ **do**
 - 2: 1. Perform Sparse Fuzzy c -Means AO to obtain optimal U, V and \mathbf{w} from eqns. (4.7), (4.8) and (4.9)
 - 3: 2. Compute $BIC_{(s)}^{FCM}(\{U_{(s)}, V_{(s)}\})$ using Theorem 4.1.
 - 4: **end for**
 - 5: Select $s := \arg \max_s BIC_{(s)}^{FCM}$, and return the corresponding $V_{(s)}, U_{(s)}, \mathbf{w}_{(s)}$.
-

4.3.2 Bayesian Information Criterion for k -Means and Fuzzy c -Means

For k -Means, we can derive the following corrected expression for the Bayesian Information Criterion (Pelleg and Moore, 2000).

Remark 4.1. *The Bayesian Information Criterion for k -Means is*

$$\begin{aligned} BIC^{KM}(\Theta) &= \sum_{j=1}^k |C_j| \log |C_j| - |X| \log |X| - \frac{p|X|}{2} \log(2\pi\hat{\sigma}^2) \\ &\quad - \frac{p}{2}(|X| - k) - \frac{k(p+1)}{2} \log |X|, \\ \text{where, } \hat{\sigma}^2 &= \frac{1}{p(|X| - k)} \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2. \end{aligned}$$

The proof of Remark 4.1 is provided in Appendix B. The expression of $BIC^{KM}(\Theta)$ is used to form a Sparse k -Means using BIC (SKM+BIC) method, outlined in Algorithm 2, where $BIC^{KM}(\Theta)$ is used to select a suitable clustering among n_s candidate clusterings.

For Fuzzy c -Means, we derive an expression for the Bayesian Information Criterion.

Theorem 4.1. *The Bayesian Information Criterion for Fuzzy c -Means is*

$$\begin{aligned} BIC^{FCM}(\Theta) &= \sum_{i=1}^n \sum_{j=1}^k \log \mu_{ij} - \frac{kp|X|}{2} \log(2\pi\hat{\sigma}^2) \\ &\quad - \frac{kp(|X| - k)}{2} - \frac{k(|X| + p)}{2} \log |X|, \\ \text{where, } \hat{\sigma}^2 &= \frac{1}{kp(|X| - k)} \sum_{i=1}^n \sum_{j=1}^k \|\mathbf{x}_i - \mathbf{v}_j\|^2. \end{aligned}$$

Proof. From the definition of BIC in (4.11), we can write the BIC for Fuzzy c -Means as,

$$BIC^{FCM}(\Theta) = l(D; \Theta) - \frac{k(|X| + p)}{2} \log |X|. \quad (4.14)$$

The number of parameters $|\Theta| = k(|X| + p)$ due to the kp cluster centers and $k|X|$ fuzzy cluster memberships. The log likelihood function is defined as,

$$\begin{aligned} l(D; \Theta) &= \log \prod_{i=1}^n P(\mathbf{x}_i) \\ &= \sum_{i=1}^n \log \sum_{j=1}^k P(\mathbf{x}_i | \mathbf{x}_i \in C_j) P(\mathbf{x}_i \in C_j) \\ &= \sum_{i=1}^n \log \sum_{j=1}^k \mu_{ij} P(\mathbf{x}_i | \mathbf{x}_i \in C_j) \\ &\geq \sum_{i=1}^n \sum_{j=1}^k [\log \mu_{ij} + \log P(\mathbf{x}_i | \mathbf{x}_i \in C_j)]. \end{aligned}$$

The inequality follows Jensen's inequality. Since all optimal μ_{ij} are obtained from Fuzzy

c -Means clustering, the equality holds, and therefore,

$$l(D; \Theta) = \sum_{i=1}^n \sum_{j=1}^k [\log \mu_{ij} + \log P(\mathbf{x}_i | \mathbf{x}_i \in C_j)].$$

The sampling probability of \mathbf{x}_i from each C_j follows the normal distribution,

$$P(\mathbf{x}_i | \mathbf{x}_i \in C_j) = \frac{1}{(2\pi\sigma^2)^{p/2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2\right).$$

where all $\sigma_j = \sigma$ since in Fuzzy c -Means we assume all spherical clusters with equal variances. Then,

$$l(D; \Theta) = \sum_{i=1}^n \sum_{j=1}^k \log \mu_{ij} - \frac{kp|X|}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n \sum_{j=1}^k \|\mathbf{x}_i - \mathbf{v}_j\|^2.$$

To estimate σ to maximize the log likelihood function, we equate the derivative to zero and solve for σ^2 ,

$$\begin{aligned} \frac{\partial}{\partial \sigma} l(D; \Theta) &= 0 \\ \implies -\frac{kp|X|}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^n \sum_{j=1}^k \|\mathbf{x}_i - \mathbf{v}_j\|^2 &= 0 \\ \implies \sigma^2 &= \frac{1}{kp|X|} \sum_{i=1}^n \sum_{j=1}^k \|\mathbf{x}_i - \mathbf{v}_j\|^2. \end{aligned}$$

The unbiased estimator of the variance of each cluster σ_j^2 can then be written as,

$$\hat{\sigma}_{jUBE}^2 = \frac{1}{kp(|C_j| - 1)} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{v}_j\|^2$$

Assuming all $\hat{\sigma}_j = \hat{\sigma}$, and summing over all clusters,

$$\begin{aligned} \sum_{j=1}^k \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{v}_j\|^2 &= kp \sum_{j=1}^k (|C_j| - 1) \hat{\sigma}_j^2 \\ \implies \sum_{j=1}^k \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{v}_j\|^2 &= kp \hat{\sigma}^2 (|X| - k) \\ \implies \hat{\sigma}^2 &= \frac{1}{kp(|X| - k)} \sum_{j=1}^k \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{v}_j\|^2. \end{aligned} \tag{4.15}$$

Substituting this estimator of σ^2 in the log likelihood function and simplifying, yields

$$l(D; \Theta) = \sum_{j=1}^k \sum_{i=1}^n \log \mu_{ij} - \frac{kp|X|}{2} \log(2\pi\hat{\sigma}^2) - \frac{kp}{2}(|X| - k).$$

Substituting in (4.14), we obtain,

$$\begin{aligned} BIC^{FCM}(\Theta) &= \sum_{j=1}^k \sum_{i=1}^n \log \mu_{ij} - \frac{kp|X|}{2} \log(2\pi\hat{\sigma}^2) \\ &\quad - \frac{kp}{2}(|X| - k) - \frac{k(|X| + p)}{2} \log |X|. \end{aligned} \tag{4.16}$$

Eqs. (4.16) and (4.15) prove Remark 4.1. □

We use the expression of BIC^{FCM} in Theorem 4.1 to form a Sparse Fuzzy c -Means using BIC (SFCM+BIC), outlined in Algorithm 3, to select an appropriate clustering from n_s candidate clusterings.

4.3.3 On Computation Complexity

Both SKM+BIC and SFCM+BIC involve AO steps that have the complexity of $O(knp)$ per iteration. Both BIC^{KM} and BIC^{FCM} involve a single $O(knp)$ computation step. Thus, clustering at all n_s candidate upper bounds leads to an overall complexity of $O(n_s knp)$ for both SKM+BIC and SFCM+BIC. This is a large reduction in complexity from the $O((B + 1)n_s knp)$ computational requirements of Sparse k -Means and Sparse Fuzzy c -Means using the Permutation Method (SKM+PM and SFCM+PM, respectively). As the size of the dataset increases, the execution times of SKM+PM and SFCM+PM increase drastically in comparison to SKM+BIC and SFCM+BIC, as shown in Figure 4.1. This further motivates us to study the clustering performance of SKM+BIC and SFCM+BIC, since they are more applicable for the sparse clustering of larger datasets.

4.4 Experiments and Results

In this section, we investigate the performance of using BIC to select the appropriate clustering from Sparse k -Means (SKM+BIC) and Sparse Fuzzy c -Means (SFCM+BIC). We first pose the question of whether a global comparison of clusterings can yield an advantage in the selection of sparse clustering models, for which we compare different approaches involving BIC as well as several cluster validity indices against the performance of local sparse clustering model selections using the Permutation Method. Next, we compare the overall performance of SKM+BIC for sparse hard clustering with Sparse k -Means using the Permutation Method (SKM+PM) and recent sparse hard clustering methods on several synthetic and real datasets. Similarly, we also compare the performance of SFCM+BIC for sparse fuzzy clustering with Sparse Fuzzy c -Means using the Permutation Method (SFCM+PM) and recent sparse fuzzy clustering methods on several synthetic and real datasets.

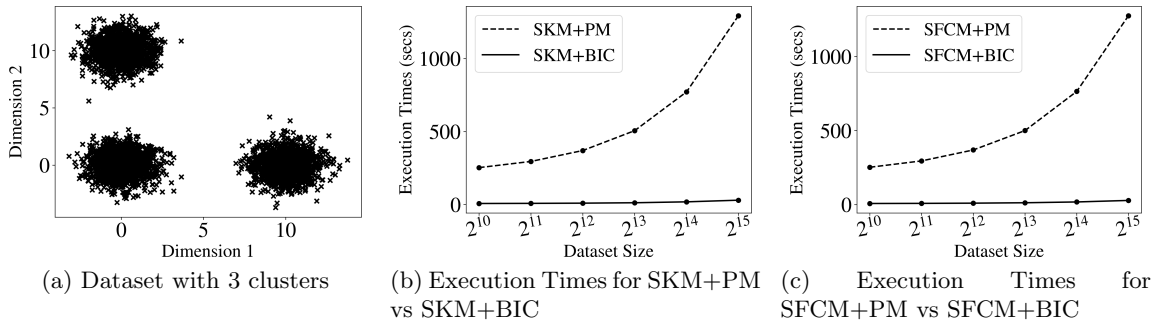


Figure 4.1: Comparison of the execution times between the proposed SKM+BIC and SKM+PM, and between the proposed SFCM+BIC and SFCM+PM. Datasets containing three clusters are generated with increasing sizes (from 2^{10} to 2^{15}) and the increase in execution times are noted. By using BIC as an alternative to the Permutation Method, SKM+BIC and SFCM+BIC have drastically lower growths in execution times. The details of the empirical comparison of timings are provided in Section B.2 of Appendix B

4.4.1 Effectiveness of global sparse clustering model selection

To test the effectiveness of approaches that perform sparse clustering model selection globally on the original data space, we compare the performance of a total of 21 approaches to sparse clustering model selection. The performance of local sparse clustering model selection by the GAP statistic used by the Permutation Method is compared with the global method of BIC, in addition to nineteen other cluster validity indices shown in Table 4.1. The cluster validity indices considered include several recent, as well as classical indices that have over time shown excellent performance in several applications (Arbelaitz et al., 2013; Luna-Romera et al., 2019). The selection criteria used for each approach and their computation complexity are listed in the table as well.

We test all approaches on a set of 44 synthetic datasets. 40 of these datasets are high dimension from the MOCK collection of datasets to benchmark clustering methods (Handl and Knowles, 2007). The datasets are ten-dimensional consisting of 40, 20, 10, or 4 clusters. Other than the MOCK datasets, there are four two-dimensional datasets containing 15, 8, 6, and 4 clusters. They are used to easily visualize that even in low dimensions the approaches perform clustering effectively.

We follow a common protocol for all our experiments. We follow the default parameter settings of the Sparcl implementation of Sparse k -Means, and consider 10 candidate upper bounds to the ℓ_1 -norm, (i.e., n_s is set to 10). On each dataset and for each candidate upper bound, a sparse clustering method is run for a maximum of 20 iterations with an error tolerance of 10^{-6} . The clustering is re-run 20 times and the clustering result with the lowest cost is selected. For the Permutation Method, 25 permuted datasets are generated and clustered along with the original dataset. For Sparse Fuzzy c -Means, m is set to 1.2. The returned clustering results are evaluated using each of the approaches in Table 4.1 and using their selection criteria, the final clustering is selected. For hard clustering methods, cluster validity indices using fuzzy cluster memberships μ_{ij} are computed from the resulting cluster centers. The final clustering is evaluated using the Adjusted Rand Index (ARI). For

Table 4.1: Specification of the Compared Cluster Validity Indices.

Sl. No.	Name	Selection Criteria	Complexity
1	Akaike Information Criteria (AIC) (Akaike, 1974)	max	$O(knp)$
2	Caliński Harabasz (CH) Index (Caliński and Harabasz, 1974)	max	$O(knp)$
3	Classification Entropy (CE) (Bezdek, 1975)	min	$O(knp)$
4	Davies Bouldin (DB) Index (Davies and Bouldin, 1979)	min	$O(k^2np)$
5	Fukuyama Sugeno (FS) Index (Fukuyama and Sugeno, 1989)	min	$O(knp)$
6	Fuzzy Hypervolume (FHV) (Dave, 1996)	min	$O(knp)$
7	I Index (Maulik and Bandyopadhyay, 2002)	max	$O(knp)$
8	Modified Partition Coefficient (MPC) (Dave, 1996)	max	$O(knp)$
9	Modified Partition Coefficient (PC) (Dave, 1996)	max	$O(knp)$
10	Partition Index (PI) (Bensaid et al., 1996)	min	$O(knp)$
11	PBMF Index (Pakhira et al., 2004)	max	$O(knp)$
12	PCAES (Wu and Yang, 2005)	max	$O(knp)$
13	RLWY Index (Ren et al., 2016)	min	$O(knp)$
14	Rezaee Index (Rez) (Rezaee, 2010)	min	$O(k^2np)$
15	Silhouette Index (SIL) (Rousseeuw, 1987)	max	$O(n^2)$
16	Xie-Beni (XB) Index (Xie and Beni, 1991)	min	$O(knp)$
17	Xu Index (Xu, 1997)	min	$O(knp)$
18	ZXF Index (Zhao et al., 2009b)	min	$O(knp)$
19	ZWZL Index (Zhang et al., 2008)	min	$O(k^2np)$
20	GAP Statistic (Tibshirani et al., 2001)	max	$O(Bknp)$
21	Bayesian Information Criteria (BIC)	max	$O(knp)$

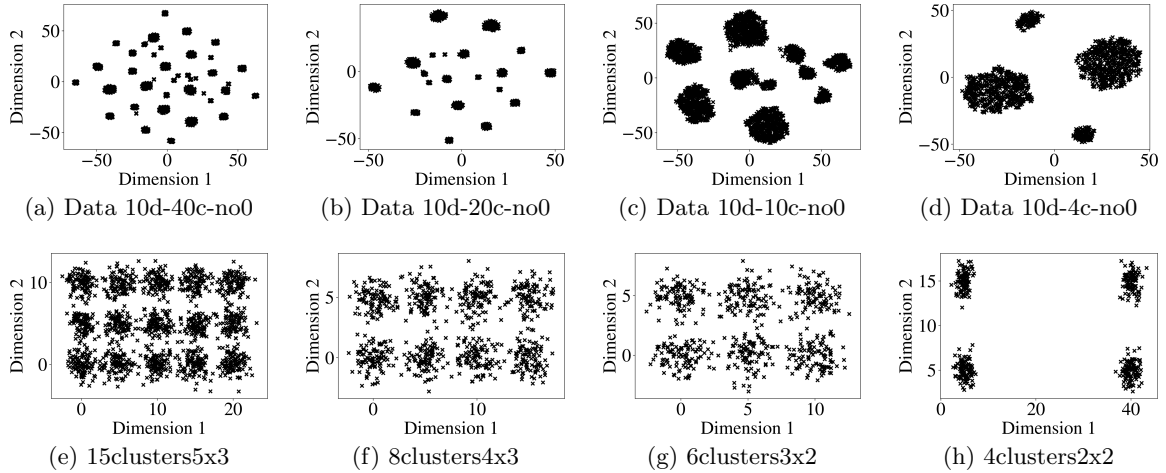


Figure 4.2: Synthetic datasets - Figures (a) to (d) are TSNE plots of ten-dimensional data from the MOCK collection of clustering datasets, consisting of (a) 40, (b) 20, (c) 10, and (d) 4 clusters. Figures (e) to (h) are two-dimensional generated data to ensure the sparse methods work in low dimensions as well.

fuzzy sparse clustering methods, the fuzzy cluster membership matrix U is used to obtain discrete cluster memberships for ARI evaluation. The entire process is repeated five times, and the average ARI obtained by the approach on that dataset is recorded. On each dataset, all approaches are ranked on decreasing ARI scores, and the average rank obtained by each method across all datasets is noted. We perform the Wilcoxon signed-rank test between

our proposed approach of using BIC and all other approaches. The null hypothesis H_0 is on whether the average ARI obtained by a pair of approaches are statistically comparable, and the alternate hypothesis H_1 is on whether the difference in obtained average ARI is statistically significant. H_0 is rejected if the p-value of the test is less than 0.5.

Table 4.2 summarises the results of the identification of the sparse hard clustering model selection across different degrees of sparsity, where Sparse k -Means is performed to obtain all hard sparse clusterings. We observe that our proposed approach of using BIC for hard clustering model selection obtains the lowest average rank among all 21 approaches considered. In combination with the results of the Wilcoxon signed-rank test, we observe that the proposed BIC approach significantly outperforms all the 20 approaches considered. We also note that the GAP statistic has obtained quite a high average rank, even though it computes the GAP statistic locally at every degree of sparsity. In comparison, we observe several cluster validity indices have obtained much lower average ranks, even though the computation of the index is global, and therefore the clusterings are evaluated and compared in the original data space. Other than BIC (1.07), we observe low average ranks in CH (2.61), CE (3.05), MPC (2.27), PC (2.27), Rez (2.91), SIL (2.39), Xu (2.39), ZXF (2.77), which have all obtained average ranks lower than that of GAP (11.32). This shows that global sparse hard clustering model selection can be effective in the original data space, with BIC being the best choice for the task. The full results are provided in Section B.4 of Appendix B.

Table 4.2: Average Rank and Wilcoxon Signed-Rank Test Results on the Average ARI achieved by the 21 Approaches for Sparse k -Means Clustering Model Selection across Different Degrees of Sparsity.

Method	AIC	CH	CE	DB	FS	FHV	I
Avg. Ranks	17.75	2.61	3.05	17.20	10.80	5.34	10.25
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-value	1.65E-08	5.61E-06	2.56E-06	2.42E-08	3.57E-08	5.39E-07	1.68E-07
Method	MPC	PC	PI	PBMF	PCAES	RLWY	Rez
Avg. Ranks	2.27	2.27	4.59	12.07	17.41	4.59	2.91
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-value	3.79E-06	3.79E-06	1.17E-06	5.25E-08	2.42E-08	1.17E-06	2.56E-06
Method	SIL	XB	Xu	ZXF	ZWZL	GAP	BIC
Avg. Ranks	2.39	7.41	2.39	2.77	10.43	11.32	1.07
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	5.61E-06	2.48E-07	5.61E-06	2.56E-06	2.48E-07	4.38E-07	-

Table 4.3 summarises the results of sparse fuzzy clustering model selection across different levels of sparsity. Here Sparse Fuzzy c -Means performs all fuzzy sparse clusterings. Our proposed approach of using BIC to identify the best fuzzy sparse clustering is observed to obtain the lowest average rank and has statistically significant performance as shown by the Wilcoxon signed-rank test, thereby significantly outperforming the other 20 approaches considered. As in the case of hard clustering, we also observe that the exact same set of cluster validity indices that compare the sparse clusterings in the original data space have also obtained lower average ranks than GAP (5.73). This can be observed with BIC(1.41), CH (3.00), CE (3.43), MPC (2.43), PC (2.43), Rez (2.94), SIL (2.02), Xu (2.75), ZXF (2.77), which have all obtained average ranks lower than that of GAP (5.73). Thus even for sparse

fuzzy clustering, the clusterings can be compared effectively in the original data space, with BIC performing the best for this task. The full results observed for all approaches are provided in Section B.4 of Appendix B.

Table 4.3: Average Rank and Wilcoxon Signed-Rank Test Results on the Average ARI achieved by the 21 approaches to Sparse Fuzzy c -Means Clustering Model Selection across Different Degrees of Sparsity.

Method	AIC	CH	CE	DB	FS	FHV	I
Avg. Ranks	18.02	3.00	3.43	17.64	13.18	11.50	9.64
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-value	1.77E-08	5.90E-06	3.79E-06	2.61E-08	5.26E-08	1.14E-07	1.94E-07
Method	MPC	PC	PI	PBMF	PCAES	RLWY	Rez
Avg. Ranks	2.43	2.43	5.52	11.86	17.73	5.52	2.93
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1
p-value	7.26E-06	7.26E-06	1.73E-06	8.71E-08	2.42E-08	1.73E-06	6.52E-06
Method	SIL	XB	Xu	ZXF	ZWZL	GAP	BIC
Avg. Ranks	2.02	7.30	2.75	2.75	6.66	5.73	1.41
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	8.07E-06	1.17E-06	8.07E-06	6.52E-06	1.34E-06	1.32E-03	-

4.4.2 Comparing sparse clustering methods on synthetic datasets

In this section, we compare the proposed approach of SKM+BIC with recent sparse hard clustering methods on synthetic datasets, followed by comparing SFM+BIC with recent sparse fuzzy clustering methods on the same set of synthetic datasets. We consider the method of Robust Sparse k -Means (RSKM) (Kondo et al., 2012) and the recent Structured Sparse k -Means (SSKM) (Gong et al., 2018) for comparison, along with the original Sparse k -Means which uses the permutation methods (SKM+PM). All methods are run on the 44 synthetic datasets following the experiment protocol discussed in Section 4.4.1. RSKM has a parameter α , which is set to 0.1. SSKM has a parameter λ which is set to 10^4 . Additionally, SSKM involves the construction of a Laplacian matrix, which we set as the adjacency matrix A constructed as the inverse of the Euclidean distance between two features, i.e., $A_{ij} = \frac{1}{\sqrt{\sum_{l=1}^n (x_{li} - x_{lj})^2}}$.

Table 4.4 summarizes the clustering performance of the sparse hard clustering methods on the synthetic datasets. Our proposed SKM+BIC is observed to attain the lowest average rank, in addition to obtaining a statistically significant average ARI as shown by the Wilcoxon signed-rank test. This shows the efficacy of the proposed approach of SKM+BIC to perform sparse hard clustering even in comparison to recent sparse hard clustering methods. The full results containing the average ARI achieved by each method are provided in Section B.5 of Appendix B.

Next, we compare the proposed SFM+BIC to the method of Robust Sparse Fuzzy c -Means (RSFMC) (Xu et al., 2016) using two norms: the $\ell_{2,1}$ -norm (RSFMC($\ell_{2,1}$)) and the ℓ_1 -capped norm (RSFMC(ℓ_1 -c)). RSFMC (Xu et al., 2016) has a regularization parameter λ which is set to 0.1, and RSFMC using the capped l_1 -norm has a threshold ϵ of the capped

Table 4.4: Average Rank and Wilcoxon Signed-Rank Test on the Average ARI achieved by the Sparse Hard Clustering Methods on the Synthetic Datasets.

Methods	SKM+PM	RSKM	SSKM	SKM+BIC
Avg. Ranks	2.25	2.48	3.93	1.18
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	4.38E-07	1.50E-07	7.62E-09	-

l_1 -norm, which is set to 1.5. Also, we compare the performance with the baseline Sparse Fuzzy c -Means using the permutation method (SFCM+PM). The methods are compared on the 44 synthetic datasets following the experimental protocol discussed in Section 4.4.1. In Table 4.5 we observe the summary of the clustering performance of the sparse fuzzy clustering methods on the synthetic datasets, with the full results containing the average ARI achieved by each method provided in Section B.5 of Appendix B. The proposed SFCM+BIC is observed to attain the lowest average rank on average ARI obtained, in addition to achieving statistically significant average ARI as shown by the Wilcoxon signed-rank test. Thus on synthetic datasets, we observe the efficacy of our proposed approach of SFCM+BIC to perform sparse fuzzy clustering in comparison to recent sparse fuzzy clustering methods.

Table 4.5: Average rank and Wilcoxon Signed-Rank Test on the Average ARI achieved by the Sparse Fuzzy Clustering Methods on the Synthetic Datasets.

Methods	SFCM+PI	RSFCM- $(l_{2,1})$	RSFCM- (l_{1c})	SFCM+BIC
Avg. Ranks	1.70	3.07	3.75	1.09
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	1.32E-03	1.12E-08	7.61E-09	-

4.4.3 Comparing sparse clustering methods on real datasets

From the results of the previous experiments, we observed how the proposed approaches of SKM+BIC and SFCM+BIC led to improved identification of sparse clusterings on synthetic datasets. In this section, we investigate the performance of the proposed approaches on real-world datasets in comparison to recent hard and fuzzy sparse clustering methods. The methods in contention are the same as were considered for the experiments on synthetic datasets in Section 4.4.2. The sparse clustering methods are evaluated on six high dimensional gene expression cancer datasets available at <http://www.stat.cmu.edu/~jiashun/Research/software/GenomicsData/>. Each of these datasets contains the patient information and a label representing the presence and absence of the types of cancer considered. The datasets have a high range in the number of features, from 2000 features for the *colon cancer* dataset to 12533 features in the *lung cancer* dataset. Besides, the clustering methods are compared on eleven real-world datasets from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017), shown in Table 4.6. Therefore the performance on these datasets reflects how the clustering methods perform feature selection as well as clustering in a real-world setting. The UCI dataset *ecoli* is preprocessed to remove three classes with less than six data instances. The *football* dataset (available at

<http://www-personal.umich.edu/~mejn/netdata/>) is a network dataset on 115 American football games played across 12 regions. The same experiment protocol is followed as outlined in Section 4.4.1, and the parameters for the contending methods were set as stated in Section 4.4.2.

Table 4.6: Specifications of the Real Datasets.

Dataset	Size (n, p)	k
brain cancer	(42, 5597)	5
colon cancer	(62, 2000)	2
lung cancer	(181, 12533)	2
lymphoma cancer	(62, 4026)	3
prostate cancer	(102, 6033)	2
srbc cancer	(63, 2308)	4
ecoli	(327, 7)	5
football	(115, 115)	12
gesture	(1743, 51)	5
libras	(360, 90)	15
seeds	(210, 7)	3
segment	(2310, 19)	7
spambase	(4601, 57)	2
ukm	(403, 5)	4
wdbc	(683, 9)	2
wine	(178, 13)	3
banknote	(1372, 4)	2
yeast	(1484, 8)	10

Table 4.7 shows the results of the sparse hard clustering methods on the real datasets. The bottom three rows of the table summarize the results, from which we note that our proposed SKM+BIC attained the lowest average rank while achieving statistically significant performance in comparison to the contending methods. Therefore we conclude that SKM+BIC has attained significantly better performance in comparison to the contending methods. We observe that SKM+BIC has attained the best results in five of the six high-dimensional gene expression cancer datasets, with very competent performance in the sixth dataset. On the UCI datasets, we observe that SKM+BIC achieves the best ARI on most datasets while attaining highly competent results on the rest. Thus the results show that SKM+BIC is a better method for the sparse hard clustering of real-world datasets, in comparison to the baseline SKM+PM as well as recent sparse clustering methods.

In Table 4.8 we observe the results of the sparse fuzzy clustering methods on the real datasets. We observe from the bottom three rows of the table that our proposed SFCM+BIC achieves the lowest average rank while attaining statistically significant performance in comparison to the contending methods, which lets us conclude that SFCM+BIC has achieved significant improvement in sparse clustering performance in comparison to the contending methods. We observe that SFCM+BIC has achieved the highest average ARI on six of the high-dimension gene expression cancer datasets. Among the UCI datasets as well we observe that SFCM+BIC either achieves the highest average ARI or has a very competing performance. Thus using BIC for sparse clustering model selection has led to SFCM+BIC being a better alternative to sparse fuzzy clustering, in comparison to the baseline SFCM+PM or the contending sparse fuzzy clustering methods.

Table 4.7: Average ARI achieved by the Sparse Hard Clustering Methods on the Real Datasets.

Methods	SKM+PM	RSKM	SSKM	SKM+BIC
brain cancer	0.5471	0.3762	0.4144	0.5975
colon cancer	0.1346	-0.0009	0.0826	0.5420
lung cancer	0.3584	0.0519	0.0087	0.3584
lymphoma cancer	0.8964	0.3666	0.6313	0.9471
prostate cancer	0.0386	0.0295	0.016	0.0386
srbc cancer	0.4080	0.1842	0.2216	0.3979
ecoli	0.4539	0.4224	-0.0251	0.5118
football	0.7341	0.4565	0.4182	0.8565
gesture	0.1714	0.153	0.02	0.1985
libras	0.2875	0.2794	0.0293	0.3056
seeds	0.6115	0.6500	0.0416	0.6115
segment	0.1241	0.1204	0.0292	0.2143
spambase	0.0381	0.1229	0.0012	0.0381
ukm	0.3601	0.3555	0.1014	0.3555
wdbc	0.8409	0.7039	0.5297	0.8409
wine	0.3711	0.3032	0.0015	0.3711
yeast	0.0706	0.0648	0.0393	0.0782
Avg. Ranks	1.65	2.94	3.76	1.24
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	0.0099	0.0032	0.0003	-

Table 4.8: Average ARI achieved by the Sparse Fuzzy Clustering Methods on the Real Datasets.

Methods	SFCM+PM	RSFCM- $(l_{2,1})$	RSFCM- (l_{1c})	SFCM+BIC
brain cancer	0.5212	0.198	0.0061	0.5842
colon cancer	0.4474	0.0111	0.0861	0.5420
lung cancer	0.3250	-0.0039	-0.022	0.3250
lymphoma cancer	0.3902	0.4818	0.3552	0.5048
prostate cancer	0.0321	0.0154	-0.0090	0.0578
srbc cancer	0.3125	0.1106	0.0181	0.3240
ecoli	0.4399	0.4670	0.0019	0.4399
football	0.0480	0.7756	0.0582	0.8547
gesture	0.1951	0.2788	0.2142	0.2113
libras	0.1360	0.1025	0.0067	0.3109
seeds	0.6206	0.7166	0.5689	0.6206
segment	0.1203	0.4134	0.4138	0.2073
spambase	0.0411	0.0898	0.0860	0.0411
ukm	0.3555	0.1953	0.1482	0.3555
wdbc	0.8355	0.7930	0.7040	0.8355
wine	0.3711	0.3407	0.2939	0.3711
yeast	0.0808	0.0982	0.1042	0.0686
Avg. Ranks	2.24	2.47	3.29	1.59
Hypothesis Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
p-value	0.0093	0.0217	0.0036	-

4.5 Discussion

In this paper, we proposed an improved sparse clustering model selection by the use of expressions of Bayesian Information Criterion (BIC) derived for k -Means and Fuzzy c -Means. Traditional sparse clustering methods have a $O(knp)$ complexity to obtain a clustering at a specific degree of sparsity, however using the Permutation Method for sparse clustering model selection from several clusterings at different degrees of sparsity increases the overall computational cost to a significantly higher $O((B+1)n_s knp)$. This is due to the Permutation Method which generates and clusters additional B permuted datasets locally at each of the n_s degrees of sparsity. We instead proposed a global sparse clustering model selection, where a suitable sparse clustering can be selected from several sparse clusterings at different degrees of sparsity. The proposed global sparse clustering model selection occurs in the original data space using our derived expressions of BIC, which requires significantly lower computational costs of $O(n_s knp)$. From experiments involving several cluster validity indices, we observed that global clustering model selection approaches showed improvements in comparison to local model selection by the Permutation Method, with BIC performing the best at the model selection. Experiments on several synthetic and real datasets showed that using BIC for sparse clustering model selection led to state-of-the-art clustering performances for both Sparse k -Means (SKM+BIC) and Sparse Fuzzy c -Means (SFCM+BIC). The low cost of computation of BIC and its improved performance in sparse clustering model selection makes the proposed SKM+BIC and SFCM+BIC the recommended approach for the sparse selection of larger datasets.

Chapter 5

Fuzzy Clustering to Identify Clusters at Different Levels of Fuzziness: An Evolutionary Multi-Objective Optimization Approach

Summary

Fuzzy clustering methods identify naturally occurring clusters in a dataset, where the extent to which different clusters are overlapped can differ. Most methods have a parameter to fix the level of fuzziness. However, the appropriate level of fuzziness depends on the application at hand. This chapter presents Entropy c -Means (ECM), a method of fuzzy clustering that simultaneously optimizes two contradictory objective functions, resulting in the creation of fuzzy clusters with different levels of fuzziness. This allows ECM to identify clusters with different degrees of overlap. ECM optimizes the two objective functions using two multi-objective optimization methods, Non-dominated Sorting Genetic Algorithm II (NSGA-II), and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D). We also propose a method to select a suitable trade-off clustering from the Pareto front. Experiments on challenging synthetic datasets as well as real-world datasets show that ECM leads to better cluster detection compared to the conventional fuzzy clustering methods as well as previously used multi-objective methods for fuzzy clustering.

5.1 Introduction

In the popular center-based clustering problem of k -Means, data instances are assigned to distinct clusters. However, the notion of clusters can be generalized in different ways, for example the Fuzzy c -Means problem (Dunn, 1973; Bezdek, 1973; Bezdek et al., 1984) uses fuzzy set theory to generalize the concept of cluster membership, so that data instances can have some degree of membership to all clusters. This motivated us to identify one of the factors affecting center-based clustering to be the nature of the clusters to be identified,

as was discussed in Section 1.2. The nature of the cluster being identified by center-based clustering problems can be of two types: hard or fuzzy. In hard center-based clustering, each data point is assigned a membership in the set $\{0, 1\}$ to all clusters, whereas in fuzzy center-based clustering (Döring et al., 2006; Peters et al., 2013), where every data point is assigned a membership in the range $[0, 1]$ to all clusters. A high membership value indicates that a data point is closer to the center of the corresponding cluster. Fuzzy center-based clustering thus generalizes the membership values from the set $\{0, 1\}$ to the interval $[0, 1]$. This generalization can be done in several ways, collectively called *fuzzification*. The degree of fuzzification has an important effect on the identification of overlap between clusters. The true overlap between clusters is unknown, since in the problem of data clustering we have no information on the underlying cluster structure. Low degrees of fuzzification assign high memberships for a data point to its closest cluster, and assign low memberships for it to the other clusters. This leads to the formation of clusters that are less overlapped. On the other hand, increasing the degree of fuzziness decreases the memberships of data instances to their closest cluster and increases their memberships to all other clusters, forming more overlapped clusters. With the appropriate degree of fuzzification, a fuzzy center-based clustering method can form fuzzy clusters that correspond to the underlying overlapped cluster structure (Klawonn and Höppner, 2003).

The proper extent of fuzzification can thus help to identify clusters with varying degrees of overlap. Moreover, the positions of the identified cluster centers are less sensitive to the presence of noise in the dataset compared to hard center-based clustering. With this aim, Dunn first introduced fuzzy center-based clustering with a specific level of fuzziness of the membership values (Dunn, 1973). This was later generalized by Bezdek (1981), who introduced a parameter m with which one can choose an appropriate level of fuzziness. This gave rise to the well-known Fuzzy c -Means (FCM) problem

$$\text{minimize } J_m = \sum_{i=1}^N \sum_{j=1}^c \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2, \quad (5.1)$$

subject to the constraints $\sum_{j=1}^c \mu_{ij} = 1$, where $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \dots, N$ are the data instances to be grouped into c clusters having centers $\mathbf{v}_j \in \mathbb{R}^d$, for $j = 1, \dots, c$. Each \mathbf{x}_i belongs to the j -th cluster with a membership degree $\mu_{ij} \in [0, 1]$. Here m is the degree of fuzzification that is to be set by the user.

The popular algorithm for FCM is a two-step Alternating Optimization (AO) method that alternately updates all μ_{ij} and then all \mathbf{v}_j , so as to minimize J_m . Bezdek et al. (1984) observed that the AO method works well for values of m in the interval $[1, 30]$, with $1.5 \leq m \leq 3.0$ yielding good results for the datasets they considered. Pal and Bezdek (1995) further refined this interval to $[1.5, 2.5]$. Some approaches exist that attempt to estimate an exact value for m . Dembélé and Kastner (2003) attempted to empirically estimate m from the pairwise distances between the data instances. Futschik and Carlisle (2005) compared the clustering results over different values of m with that of random data to select one appropriate value of m from a discrete set of possible values. Schwämmle and Jensen (2010) provided an empirical formula to estimate m by considering that the probability of a cluster being well-defined decreases exponentially as the dimension of the dataset increases, while decreasing at a slightly slower rate as the number of data instances increases. Ozkan and Turksen (2007) recommended the interval $[1.4, 2.6]$ as an effective range to be used in practice.

A number of theoretical approaches to estimate suitable intervals for m exist as well. Yu et al. (2004) derived theoretical rules that are dataset dependent to estimate a range of values for m . Huang et al. (2012b) provided an alternative theoretical approach to provide a different interval for the values of m . Wu (2012) further extended the work by Yu et al. (2004) to recommend an interval of $[1.5, 4.0]$ for m based on a robust analysis of FCM. Some approaches that obtain fuzzy clusterings by eliminating the m include a semi-nonnegative matrix factorization based approach (Suleman, 2015) where cluster centers are obtained as a convex combination of the data instances. Yang and Nataliani (2017) bypasses the use of m by extending the FCM objective function to include the entropy of cluster memberships along with additional cluster prior variables. However, these approaches do not explore the full range of possible levels of fuzziness to identify an appropriate level of fuzziness suitable for the data at hand.

An interesting study by Li and Mukaidono (1995) formulated a Maximum Entropy Inference (MEI) approach to fuzzy center-based clustering,

$$\text{maximize } E = - \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \log(\mu_{ij}) , \quad (5.2)$$

subject to the constraint $\sum_{j=1}^c \mu_{ij} = 1$ and the soft constraint $\sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|^2 = 0$. This formulation allowed them to design an AO algorithm for MEI which uses an *admissible error radius* σ instead of m . This is advantageous as σ is easier to interpret compared to m . Increasing σ leads to a physical increase in the spread of the clusters, thereby increasing the fuzziness of the membership values by allowing more overlap between the clusters. Klawonn and Höppner (2003) also showed that a function of the fuzzy memberships could be used in place of u_{ij}^m in the FCM objective to control the level of fuzzification. A recent work by Saha and Das (2017) proposed an axiomatic definition of a general class of weighting functions that can be used to control the level of fuzzification. Hence based on the literature, the level of fuzzification can be set using one of the following ways (i) using an empirical formula (ii) using a parameter m whose values are to be tested from intervals derived theoretically (iii) using a function whose possible values must be controlled, (iv) using an entirely different approach as in Suleman (2015); Yang and Nataliani (2017). However, an unsupervised method that automatically decides an appropriate level of fuzzification while fitting fuzzy clusters to a dataset does not exist.

In this chapter, we show that clusterings at different levels of fuzziness can be obtained by simultaneously optimizing two contradicting objectives in a Multi-Objective Optimization (MOO) framework. In the proposed Entropy c -Means (ECM) method, one objective favors the formation of compact fuzzy clusters while the other objective favours largely overlapping clusters. In an MOO framework, multiple objective functions are optimized simultaneously, leading to a set of trade-off solutions called *Pareto-optimal* solutions, where no solution is better than the other when considering all objective function values at once (see Section 5.2.1). MOO methods have long been used for data clustering (Mukhopadhyay et al., 2014; Xia et al., 2013; Wang et al., 2013). If the objectives are contradictory, then a wide variety of Pareto-optimal solutions can be found. We show that the two objectives of ECM are indeed in conflict and hold a strong Pareto relation. This leads to a wide variety of Pareto-

optimal clusterings corresponding to different levels of fuzziness. In Section 5.3, we conduct experiments to compare the proposed ECM method against state-of-the-art fuzzy center-based clustering methods. We also propose a method to select a trade-off clustering from the Pareto-front identified by ECM in section 5.3.4.

5.2 Fuzzy Clustering using Multi-Objective Optimization

In this section we first define the framework of MOO problems, after which we elucidate the two objective functions which are optimized in ECM within an MOO framework to generate clusterings with different levels of fuzziness. Finally, we describe how the ECM problem can be solved by using two popular MOO methods, namely Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002), and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) (Zhang and Li, 2007; Zhang et al., 2009; Chen and Zhou, 2019).

5.2.1 Multi-Objective Optimization

Let there be n objective functions to be optimized, subject to p constraint functions. Problems of this form can be written as,

$$\begin{aligned} & \text{maximize/minimize} && f_i(\mathbf{x}) \quad , \quad i = 1, 2, \dots, n \\ & \text{subject to} && g_j(\mathbf{x}) \leq 0 \quad , \quad j = 1, 2, \dots, p, \end{aligned} \quad (5.3)$$

where \mathbf{x} is a feasible solution satisfying all g_j . \mathbf{x} is said to dominate \mathbf{y} , if for all i , $f_i(\mathbf{x})$ is at least as optimal as $f_i(\mathbf{y})$, and there exists at least one $f_i(\mathbf{x})$ more optimal than the corresponding $f_i(\mathbf{y})$. A feasible solution \mathbf{x}^* is said to be *optimal*, if for all feasible \mathbf{y} , \mathbf{x}^* dominates \mathbf{y} . In MOO problems, we usually cannot find single optimal solutions as the objectives are contradictory in nature. Instead we find a set of mutually non-dominating solutions called *Pareto-optimal solutions*. The set of images of the Pareto-optimal solutions in the objective space is called the *Pareto front*. For a survey on evolutionary computing approaches used to solve MOO problems, see Zhou et al. (2011).

5.2.2 Objective Functions for Fuzzy Clusters

In ECM, we propose the following two objective functions to be optimized in an MOO framework,

$$\text{minimize } f_1 = \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|^2, \quad (5.4a)$$

$$\text{maximize } f_2 = - \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \log(\mu_{ij}). \quad (5.4b)$$

One aim of center-based clustering methods is to place the cluster centers in areas having a high density of data instances. Minimizing the sum of cluster-wise *variances* ensures the formation of compact clusters. The objective function f_1 given in (5.4a) generalizes the

idea of variance by weighting the distance between data instances and cluster centers with the membership degrees of data instances in each cluster. Thus the objective of *cluster compactness* is to minimize the function f_1 to form compact clusters.

We propose the use of the objective function f_2 in (5.4b) to avoid a fixed level of fuzzification (specified by m in FCM). f_2 is the *entropy* of membership values (Li and Mukaidono, 1995), which is maximized when all membership values are equal to $1/c$, where c is the number of clusters. If a data point is close to a cluster center, the corresponding membership value will be greater than $1/c$. On the other hand, if a data point is far from a specific cluster center, its membership to that cluster will be less than $1/c$. Increasing the entropy brings all the membership values for a data point close to $1/c$. This essentially increases the level of fuzzification of the membership values.

Remark. *The two objectives to minimize f_1 and to maximize f_2 in (5.4) are contradicting.*

This can be observed easily. To minimize f_1 , for all i we set $\mu_{ij} = 1$ if $d_{ij} = \|\mathbf{x}_i - \mathbf{v}_j\|$ is minimum, and 0 for all other j . However, for such values of μ_{ij} we get $f_2 = 0$, which is the minimum value of f_2 . Thus minimizing f_1 does not maximize f_2 .

To maximize f_2 , we first form the Lagrangian for the maximization of f_2 , subject to the constraint $\sum_{j=1}^c \mu_{ij} = 1$:

$$\mathbb{L} = - \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \log(\mu_{ij}) + \sum_{i=1}^N \lambda_i \left(\sum_{j=1}^c \mu_{ij} - 1 \right). \quad (5.5)$$

From the derivative of the Lagrangian, with respect to μ_{ij} and λ_i , we get $\mu_{ij} = 1/c$. This value of μ_{ij} maximizes f_2 . However, as already observed, f_1 is minimized when $\mu_{ij} = 1$ if d_{ij} is minimum, and 0 for all other j . Hence, the values of μ_{ij} that maximize f_2 do not minimize f_1 .

Thus the above remark shows that the two objectives of minimizing f_1 and maximizing f_2 are contradicting. Optimizing both in an MOO framework leads to a Pareto-optimal set of solutions representing the best compromises between the objectives. A wide Pareto front is formed due to the contradictory nature of the objectives, identifying clusterings with different levels of fuzziness. Figure 5.1 (a) shows a synthetic dataset containing three clusters A , B , and C , where A and B are close compared to C . The Pareto front obtained by solving ECM using the NSGA-II algorithm (to be discussed in Section 5.2.3.1) is shown in Figure 5.1 (b). The clustering at end cI of the front has a minimum value of f_1 , with the estimated clusters being compact having little overlap as shown in Figure 5.1 (c). As we move towards the end cIV along the front, the clusterings become increasingly fuzzy due to the increase in f_2 (and a consequent increase in f_1), characterized by increasing overlap between the estimated clusters. As fuzziness increases from cI to cIV, the estimated clusters corresponding to closer clusters A and B first become overlapped before getting overlapped with that of C , as expected (see Figs. 5.1 (d) - 5.1 (f)).

A simple alternative might be to run FCM for different values of m . However, the AO algorithm results in erroneous solutions for higher values of m . As shown in Figure 5.2, for higher values of m , there are data instances that are assigned equal probability to all clusters. They do not contribute to the location of cluster centers, leading to convergence close to the initial center locations. The multi-objective method of ECM avoids this error. In Figure 5.1 (f) we observe that in the end of high fuzzification shown in clustering cIV, all the centers

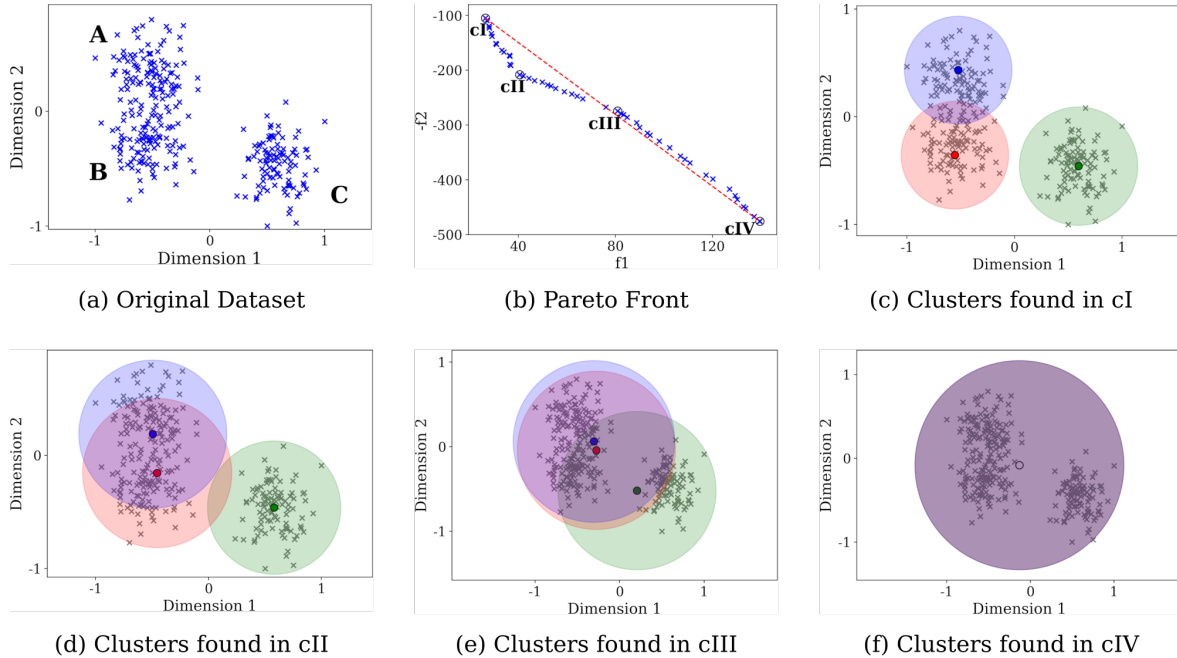


Figure 5.1: Clustering a synthetic dataset (a) using ECM-NSGA-II. The dataset contains two overlapped clusters **A** and **B**, and a third well-separated cluster **C**. At the top left corner of the Pareto front in (b), f_1 is minimized creating compact clusters with low overlap (c). At the bottom right corner, f_2 is maximized by minimizing $-f_2$, leading to more overlapped clusters as shown in (f). Across the Pareto front, clusters formed have different levels of fuzziness, see (d) and (e).

converge to the center of the dataset, as should occur at high levels of fuzzification due to the equal contribution of every point to all clusters. Another close attempt by Maximum Entropy Clustering Algorithm (MECA) (Karayiannis, 1994) identifies crisp clusters using a linear combination of the cluster compactness and the entropy of memberships, weighted by a fuzzification parameter α . MECA starts with α close to 1 where it considers maximum entropy, and over the iterations α goes close to 0 to identify crisp clusters at convergence. However, MECA does not have a stopping criteria to identify fuzzy clusters. MECA also requires specifying initial and final values to the fuzzification parameter, as well as how to decrease its value over the number of iterations. In general, one can note a renewed interest in the possible uses of the entropy of cluster memberships (bin Zhi et al., 2013; Choy et al., 2017; Yang and Nataliani, 2017). The merit of our method of ECM is that by optimizing both objectives in a multi-objective setting, ECM identifies Pareto-optimal fuzzy clusters at different levels of fuzziness, from which an optimal fuzzy clustering can be selected for the application at hand. Further demonstrations on datasets containing clusters with different degrees of overlap are discussed in Section C.2 of Appendix C. We also discuss the benefits of using multiple contradictory objectives in comparison to a combined single objective in Section C.3 of Appendix C.

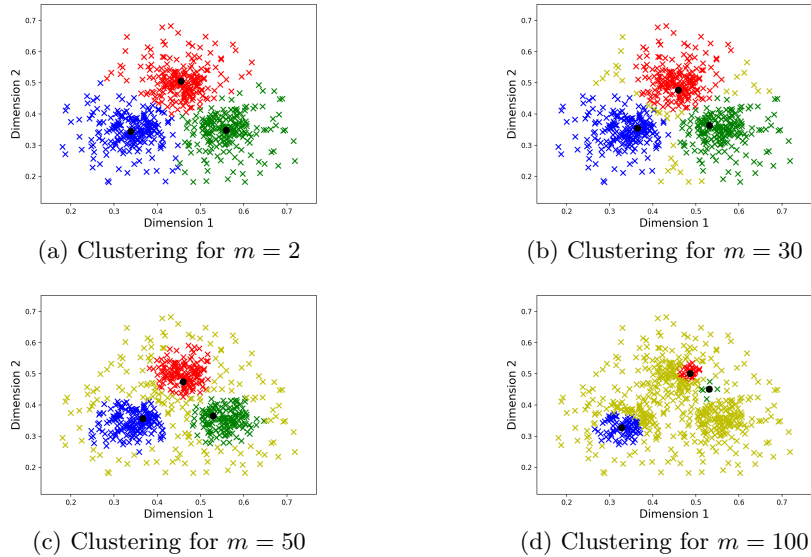


Figure 5.2: Clustering a dataset with 3 clusters for different levels of fuzziness $m = 2, 30, 50,$ and 100 . The instances in different clusters are drawn in red, blue and green and the instances in the regions of overlap between the clusters are drawn in yellow. For higher values of m , a large number of instances in the regions of overlap do not contribute to the location of cluster centers.

5.2.3 Multi-Objective Optimization Methods for ECM

There exists a wide literature on methods that can be used for the simultaneous optimization of multiple objective functions (Zhan et al., 2021; Zhou et al., 2011). Given the variety in the approaches towards multi-objective optimization, the methods may have an influence on different aspects of multi-objective clustering in different ways. We consider two methods: NSGA-II (Deb et al., 2002) and MOEA/D (Zhang and Li, 2007; Zhang et al., 2009), which have very different approaches towards multi-objective optimization. The study of both methods may reveal the benefits one may obtain when using these methods as well as other multi-objective optimization methods that are similar in their behaviour (Saini and Saha, 2021). Therefore here we discuss the use of NSGA-II and MOEA/D in the context of ECM and subsequently study their performance in detail.

5.2.3.1 ECM-NSGA-II

In each iteration of ECM-NSGA-II, a set of chromosomes is maintained called a *population*, so that multiple possible clusterings can be considered simultaneously. For each chromosome, the membership values can be computed using the centers according to the update expression derived from the Lagrangian of the cost function (5.2) of MEI:

$$\mu_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}} / \sum_{j=1}^c e^{-\frac{d_{ij}^2}{2\sigma^2}} . \quad (5.6)$$

The operations of *crossover* or *mutation* are performed to search for chromosomes yielding better objective function values. In crossover, two chromosomes can be combined to form two new chromosomes. In mutation, a single value in a chromosome can be altered. Chromosomes in a population are assigned ranks using the *non-dominated sorting*, where all chromosomes in a Pareto front are assigned the same rank. Each chromosome on a Pareto front is assigned a *crowding distance*, which is high for chromosomes with fewer chromosomes around it. *Elitist* selection of the chromosomes is undertaken for the next iteration using a *binary tournament selection*, where (i) higher ranked chromosomes have a higher probability of selection, and (ii) between chromosomes with the same rank, those with higher crowding distance have a higher probability of getting selected. A pseudo-code for ECM-NSGA-II is given in Algorithm 4.

Algorithm 4 The algorithm for ECM-NSGA-II

Input: Number of clusters c ; Population size pop ; number of function evaluations (FE).

Output: A set of pop non-dominated clusterings.

- 1: Randomly initialize all pop chromosomes as vectors of c candidate cluster centers.
 - 2: **repeat**
 - 3: **for** each chromosome **do**
 - 4: Evaluate the two objective functions in (5.4).
 - 5: Perform non-dominated sorting to compute the rank.
 - 6: Compute the crowding distance.
 - 7: **end for**
 - 8: Perform crossover and mutation to generate a new population.
 - 9: Use binary tournament selection to select pop chromosomes from the previous and new population, for the next iteration.
 - 10: **until** FE function evaluations are performed.
-

5.2.3.2 ECM-MOEA/D

ECM-MOEA/D decomposes an MOO problem with n objective functions into pop number of single objective optimization problems by using the Tchebycheff approach (Zhang and Li, 2007). In each iteration, the population contains the pop best solutions that have been found. Each subproblem is then optimized using information from its neighbouring subproblems. *Elitism* is maintained by periodically adding newly generated non-dominated solutions to an *External Population* (EP) and discarding solutions from it that are no longer non-dominated. A pseudo-code for ECM-MOEA/D is provided in Algorithm 5.

5.2.4 Computation Complexity of ECM

The time complexity of NSGA-II is $O(n(pop)^2)$ (Deb et al., 2002), whereas that of MOEA/D is $O(n(pop)T)$ (Zhang and Li, 2007). The evaluation of both objective functions requires the computation of the memberships of N data instances to c clusters. This takes $O(Nc)$ time. As $n = 2$ for ECM, every iteration of ECM-NSGA-II takes overall $O(\max\{2(pop)^2, Nc\})$ time, and that of ECM-MOEA/D takes $O(\max\{2(pop)T, Nc\})$ time. In real-world scenarios, usually $N \gg c$ as well as $N \gg (pop)^2$, leading to an $O(Nc)$ time complexity per iteration for both variants of ECM.

Algorithm 5 The algorithm for ECM-MOEA/D

Input: Number of clusters c ; number of subproblems pop ; coefficient vectors with uniform spread $\gamma_1, \dots, \gamma_{pop}$ where $\gamma_i = (\frac{i-1}{pop-1}, \frac{pop-i}{pop-1})$; number of neighbours T ; number of function evaluations FE .

Output: The external population (EP) containing non-dominated clusterings.

1: **Initialization :**

- 2: Find the T nearest neighbours for each coefficient vector.
- 3: Initialize and evaluate $\mathbf{x}_1, \dots, \mathbf{x}_{pop}$ solution vectors, each formed of c cluster centers.
- 4: Evaluate the two objective functions in (5.4) on all \mathbf{x}_i ; store the best values as z_1 and z_2 .

5: **Iteration :**

6: **repeat**

7: **for** Each coefficient vector γ_i **do**

8: Randomly select two of its T neighbours γ_j, γ_l ; retrieve the corresponding $\mathbf{x}_j, \mathbf{x}_l$.

9: Apply Differential Evolution mutation and crossover (Zhang et al., 2009) on $\mathbf{x}_i, \mathbf{x}_j$ and \mathbf{x}_l to form a new solution y .

10: If $f_1(y) < z_1$ (and/or $f_2(y) > z_2$), then update z_1 (and/or z_2).

11: If y improves upon the current best solution of γ_i or any of its T neighbours, then replace the corresponding solution with y .

12: Remove all vectors from the EP that are dominated by y , and add y to the EP if none of the existing members dominate it.

13: **end for**

14: **until** FE function evaluations are performed.

5.3 Experiments and Results

We conduct various experiments to evaluate the clustering performance of ECM-NSGA-II and ECM-MOEA/D¹. For convenience of implementation, we minimize $-f_2$ instead of maximizing f_2 as per (5.4b). The clustering performances of ECM-NSGA-II and ECM-MOEA/D are compared with the conventional AO algorithms for FCM and MEI. Further comparison is undertaken with the Multi-Objective Genetic Algorithm (MOGA) (Mukhopadhyay et al., 2006) and MOGA based Support Vector Machines (MOGA-SVM) (Mukhopadhyay and Maulik, 2009), which are two existing MOO methods used to find clusterings when the number of clusters is known. Both MOGA and MOGA-SVM use J_2 (J_m as in eqn. (5.1) with $m = 2$) and the Xie-Beni index (Xie and Beni, 1991) to find a fixed number of clusters. However, since the Xie-Beni index simply scales J_m by the distance between the closest pair of cluster centers, these two functions are not generally contradicting. Hence, the use of these two objectives is not likely to give rise to a large variety of trade-off solutions (see Figure 5.5).

In Section 5.3.1 and 5.3.2, we compare the performance of all methods on synthetic and real datasets. We also propose a method to select a clustering from the Pareto Front of ECM in Section 5.3.4. In section 5.3.3 we compare the Pareto fronts obtained by the contending MOO methods.

¹MATLAB implementations available at <https://github.com/Avisek20/ecm>.

5.3.1 Synthetic Datasets

We created 10 synthetic datasets to compare the performances of all fuzzy clustering methods. The *proximity1-5* datasets test the clustering performance when clusters are drawn closer together, as shown in Figure 5.3. Datasets *spread1-5*, on the other hand, test the performance when the spread of one cluster is progressively increased, as illustrated in Figure 5.4. Table 5.1 contains all information necessary to generate the synthetic *proximity* and *spread* datasets. We also use 20 challenging datasets from the MOCK collection² (Handl and Knowles, 2007), having either 2 or 10 dimensions. The full specifications of the synthetic datasets are present in Table 5.2.

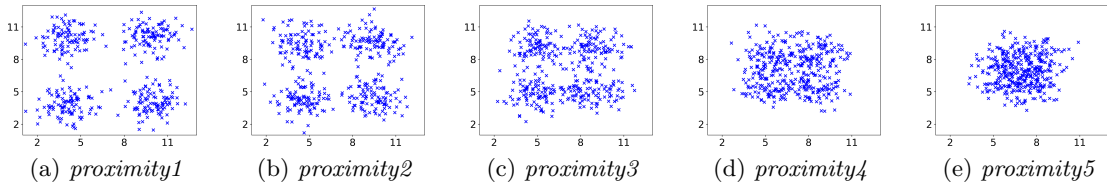


Figure 5.3: The synthetic *proximity* datasets.

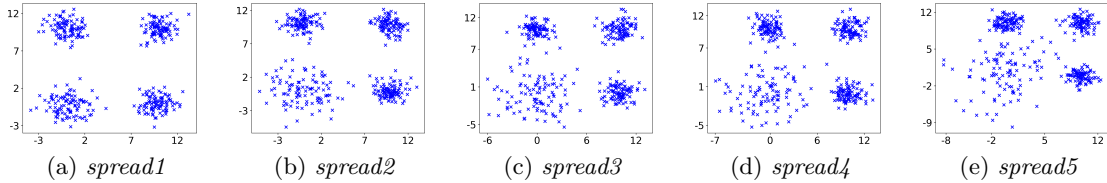


Figure 5.4: The synthetic *spread* datasets.

Table 5.1: Information to generate the Synthetic Datasets: The Number of Clusters c , Dimension of the Data dim , Number of Data Instances in each Cluster N_i , the Multidimensional Normal Distribution $N(\mu, \sigma)$, where μ is the Vector of Cluster Centers and σ is the Vector of the Standard Deviations in each Dimension.

Dataset	c	dim	N_i	Clusters
<i>proximity1</i>	4	2	100	$N([4, 4], [1, 1])$, $N([4, 10], [1, 1])$, $N([10, 4], [1, 1])$, $N([10, 10], [1, 1])$
<i>proximity2</i>	4	2	100	$N([4.5, 4.5], [1, 1])$, $N([4.5, 9.5], [1, 1])$, $N([9.5, 4.5], [1, 1])$, $N([9.5, 9.5], [1, 1])$
<i>proximity3</i>	4	2	100	$N([5, 5], [1, 1])$, $N([5, 9], [1, 1])$, $N([9, 5], [1, 1])$, $N([9, 9], [1, 1])$
<i>proximity3</i>	4	2	100	$N([5.5, 5.5], [1, 1])$, $N([5.5, 8.5], [1, 1])$, $N([8.5, 5.5], [1, 1])$, $N([8.5, 8.5], [1, 1])$
<i>proximity5</i>	4	2	100	$N([6, 6], [1, 1])$, $N([6, 8], [1, 1])$, $N([8, 6], [1, 1])$, $N([8, 8], [1, 1])$
<i>spread1</i>	4	2	100	$N([0, 0], [1, 1])$, $N([0, 10], [1, 1])$, $N([10, 0], [1, 1])$, $N([10, 10], [1, 1])$
<i>spread2</i>	4	2	100	$N([0, 0], [1.5, 1.5])$, $N([0, 10], [1, 1])$, $N([10, 0], [1, 1])$, $N([10, 10], [1, 1])$
<i>spread3</i>	4	2	100	$N([0, 0], [2, 2])$, $N([0, 10], [1, 1])$, $N([10, 0], [1, 1])$, $N([10, 10], [1, 1])$
<i>spread4</i>	4	2	100	$N([0, 0], [2.5, 2.5])$, $N([0, 10], [1, 1])$, $N([10, 0], [1, 1])$, $N([10, 10], [1, 1])$
<i>spread5</i>	4	2	100	$N([0, 0], [3, 3])$, $N([0, 10], [1, 1])$, $N([10, 0], [1, 1])$, $N([10, 10], [1, 1])$

²Available at <http://personalpages.manchester.ac.uk/mbs/julia.handl/generators.html> (last accessed November 12, 2021).

Table 5.2: Specifications of the Synthetic Datasets.

Dataset	c	Dataset Size
proximity1	4	(400,2)
proximity2	4	(400,2)
proximity3	4	(400,2)
proximity4	4	(400,2)
proximity5	4	(400,2)
spread1	4	(400,2)
spread2	4	(400,2)
spread3	4	(400,2)
spread4	4	(400,2)
spread5	4	(400,2)
2d-4c-no0	4	(1572,2)
2d-4c-no1	4	(1623,2)
2d-4c-no2	4	(1064,2)
2d-4c-no3	4	(1123,2)
2d-4c-no4	4	(863,2)
2d-4c-no5	4	(1638,2)
2d-4c-no6	4	(1670,2)
2d-4c-no7	4	(1028,2)
2d-4c-no8	4	(1078,2)
2d-4c-no9	4	(876,2)
10d-4c-no0	10	(1289,10)
10d-4c-no1	10	(958,10)
10d-4c-no2	10	(838,10)
10d-4c-no3	10	(1318,10)
10d-4c-no4	10	(933,10)
10d-4c-no5	10	(1139,10)
10d-4c-no6	10	(977,10)
10d-4c-no7	10	(1482,10)
10d-4c-no8	10	(966,10)
10d-4c-no9	10	(1183,10)

For each dataset, we normalize the data so that each feature is scaled within the range $[-1, 1]$. We run both FCM and MEI 50 times for a maximum of 5000 iterations, with an error tolerance of 10^{-16} . We run ECM-NSGA-II, ECM-MOEA/D, MOGA, and MOGA-SVM with a population size of 50 for 5000 fitness evaluations. For ECM-NSGA-II, the percentage of the population undergoing genetic operations was set to 50%, the distribution indices for crossover and mutation were both set to 20, and during tournament selection one solution is selected from 2 solutions. For ECM-MOEA/D, the size of the neighborhood is set to 50, and the crossover probability and differential weight parameters are both set to 0.5 respectively. The tuning of the parameters of the evolutionary methods to obtain these values are present in Section C.1 of Appendix C.

For FCM, we set $m = 2$, as per convention. MEI, ECM-NSGA-II and ECM-MOEA/D use the admissible error radius parameter σ . In Li and Mukaidono (1995), σ was set to 0.7. However, we have not found any general recommendations regarding the choice σ . Moreover, we have observed that the centers tend to converge if σ is large when compared to the spread of the data. We therefore decided against using a constant value of σ across all datasets. Instead for each dataset, we set σ as the standard deviation of the squared Euclidean distances between the data instances and the mean of the dataset.

In the presence of the original cluster labels, we can use the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) (calculated by assigning each data point to the cluster to which it has the maximum membership) to compare the performance of the contending methods.

For each MOO method, the clustering having maximum ARI is selected in each run and the mean of the maximum ARI over 50 runs is used for comparison. For FCM and MEI, we report the mean ARI over the 50 runs. The results are detailed in Table 5.3. The outcomes of the Wilcoxon signed-rank test³ conducted with ECM-NSGA-II and ECM-MOEA/D as the control are also listed, alongside the average ranks.

We observe that overall ECM-NSGA-II achieves the best ARI, which is also supported by the average rank as well as the signed-rank test. From Table 5.3 we can observe that on the five *proximity* datasets illustrated in Figure 5.3 and on the five *spread* datasets illustrated in Figure 5.4, ECM-NSGA-II generally achieves higher ARI values as can be observed from Table 5.3. We also observe that for the ten-dimensional data, ECM-MOEA/D generally produces higher ARI compared to the other methods. However both ECM-NSGA-II and ECM-MOEA/D generally closely follow each other in performance, as can be observed by their results which are statistically comparable. The performance of ECM-NSGA-II and ECM-MOEA/D are however significantly better compared to the performance of FCM, MEI, MOGA and MOGA-SVM.

5.3.2 Real Datasets

We compare the performance of the six methods on fifteen real datasets shown in Table 5.4. Twelve of the datasets are from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017). Three of the datasets are high-dimensional gene expression datasets, available at <http://www.stat.cmu.edu/~jiashun/Research/software/GenomicsData/>. We undertake the preprocessing of the datasets as per the discussion in Section 5.3.1 and also retain the same parameter settings for the contending methods. We report the average of the maximum ARI achieved by each method over the 50 runs in Table 5.5, along with average ranks and the results of the Wilcoxon signed-rank test. We observe that ECM-NSGA-II and ECM-MOEA/D respectively achieve the best ARI for six and nine out of the fifteen datasets. The rest of the methods do not achieve the best ARI for any of the datasets. ECM-NSGA-II and ECM-MOEA/D achieve the lowest ranks with results that are significantly different compared to the results obtained by FCM, MEI, MOGA, and MOGA-SVM. This suggests the efficacy of the ECM formulation in general and ECM-NSGA-II in particular.

5.3.3 Comparison of Pareto Fronts

Schott’s Spacing Metric (SSM) (Lizárraga et al., 2008) is a measure of the diversity of the generated Pareto-optimal solutions. We use SSM to compare between the Pareto fronts of ECM-NSGA-II and ECM-MOEA/D and the set of solutions of MOGA and MOGA-SVM mapped to the same objective space. ECM-NSGA-II and ECM-MOEA/D obtain much higher values of SSM when compared with MOGA and MOGA-SVM, as detailed in Table 5.6 which shows the average SSM observed over 50 runs. This attests to the fact that the objectives of MOGA and MOGA-SVM with a weak Pareto relation do not lend much diversity in terms of fuzziness. The entire set of solutions of MOGA and MOGA-SVM maps around the same point for the *proximity1* dataset, as seen in Figure 5.5. The higher values for ECM-MOEA/D

³ H_1 : Significantly different from the control.
 H_0 : Statistically comparable to the control.

Table 5.3: Comparison of ARI over Synthetic Datasets.

Dataset	FCM	MEI	MOGA	MOGA-SVM	ECM-NSGA-II	ECM-MOEA/D
proximity1	1.0000	0.8138	1.0000	1.0000	1.0000	1.0000
proximity2	1.0000	0.8884	1.0000	1.0000	1.0000	1.0000
proximity3	1.0000	0.9255	1.0000	1.0000	1.0000	1.0000
proximity4	1.0000	0.8868	1.0000	1.0000	1.0000	1.0000
proximity5	0.8775	0.8775	0.8812	0.8831	0.8894	0.8856
spread1	1.0000	0.9256	1.0000	1.0000	1.0000	1.0000
spread2	1.0000	0.9257	1.0000	1.0000	1.0000	1.0000
spread3	0.9867	0.9199	0.9867	0.9867	0.9893	0.9880
spread4	0.9543	0.8551	0.9543	0.9543	0.9730	0.9678
spread5	0.9479	0.8952	0.9486	0.9479	0.9632	0.9613
2d-4c-no0	0.8834	0.8691	0.8837	0.8834	0.9374	0.9050
2d-4c-no1	0.7997	0.7623	0.7730	0.7748	0.8720	0.8865
2d-4c-no2	0.7766	0.7823	0.8916	0.8909	0.9032	0.8927
2d-4c-no3	0.8333	0.8457	0.9378	0.9378	0.9405	0.9394
2d-4c-no4	0.7297	0.3147	0.7882	0.7880	0.7900	0.7785
2d-4c-no5	0.8557	0.8438	0.9020	0.9046	0.9383	0.9188
2d-4c-no6	0.9547	0.9217	0.9550	0.9552	0.9744	0.9586
2d-4c-no7	0.6424	0.6661	0.6857	0.6839	0.8429	0.8942
2d-4c-no8	0.9000	0.8792	0.9235	0.9234	0.9559	0.9242
2d-4c-no9	0.7814	0.7472	0.8888	0.8890	0.9157	0.8874
10d-4c-no0	0.7967	0.8588	0.7799	0.7891	0.8631	0.9243
10d-4c-no1	0.9797	0.9272	0.9702	0.9795	0.9781	0.9872
10d-4c-no2	0.8844	0.9168	0.8846	0.8934	0.9005	0.9119
10d-4c-no3	0.8875	0.8797	0.8881	0.8878	0.8382	0.8940
10d-4c-no4	0.8121	0.8080	0.8319	0.8142	0.8299	0.8346
10d-4c-no5	0.7178	0.7498	0.7453	0.7399	0.7554	0.8156
10d-4c-no6	0.8727	0.8670	0.8653	0.8590	0.8763	0.8746
10d-4c-no7	0.9940	0.9940	0.9940	0.9940	0.9951	0.9945
10d-4c-no8	0.9603	0.9334	0.9640	0.9639	0.9720	0.9695
10d-4c-no9	0.9577	0.8474	0.9571	0.9526	0.9176	0.9607
Avg. Rank	3.70	5.30	3.00	3.13	1.67	1.67
ECM-NSGA-II ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0
ECM-NSGA-II (p-val)	5.45E-04	7.69E-06	2.03E-03	1.84E-03	-	0.9090
ECM-MOEA/D ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-
ECM-MOEA/D (p-val)	1.82E-05	2.12E-06	1.81E-04	1.53E-04	0.9090	-

Table 5.4: Specifications of the Real Datasets.

Dataset	Dataset Size	Number of clusters
Balance Scale (B. Scale)	(625,4)	3
Breast Tissue (B. Tissue)	(106,9)	6
Breast Cancer Wisconsin (wdbc)	(683,9)	2
banknote authentication (banknote)	(1372,4)	2
Echocardiogram (echo)	(106,9)	2
Ecoli	(336,7)	8
Iris	(150,4)	3
magic	(19020,10)	2
seeds	(210,7)	3
sonar	(208,60)	2
User Knowledge Modeling (ukm)	(258,5)	4
wine	(178,13)	4
colon cancer	(62,2000)	2
lung cancer	(181,12533)	2
prostate cancer	(102,6033)	2

Table 5.5: Comparison of ARI over Real Datasets.

Dataset	FCM	MEI	MOGA	MOGA-SVM	ECM-NSGA-II	ECM-MOEA/D
B. scale	0.1448	0.1440	0.2886	0.2454	0.3156	0.2931
B. Tissue	0.2885	0.2268	0.2764	0.2763	0.2914	0.2980
wdbc	0.8300	0.8010	0.8300	0.8300	0.8831	0.8141
banknote	0.0452	0.0565	0.0925	0.0976	0.3083	0.2828
echo	0.0854	0.1371	0.0854	0.0854	0.1378	0.1342
Ecoli	0.3684	0.4225	0.4837	0.4621	0.6050	0.4591
Iris	0.7287	0.7652	0.7484	0.7409	0.8094	0.8123
magic	0.0577	0.0275	0.0758	0.0741	0.1367	0.0879
seeds	0.7266	0.4565	0.7266	0.7266	0.6798	0.7308
sonar	0.0064	0.0057	0.0100	0.0147	0.0221	0.0318
ukm	0.1777	0.3227	0.2058	0.2188	0.2476	0.3428
wine	0.8498	0.3786	0.8666	0.8649	0.8511	0.8743
colon cancer	-0.0064	0.0310	0.0122	0.0332	0.0510	0.3089
lung cancer	-0.0003	-0.0093	0.0384	0.0416	0.0413	0.3090
prostate cancer	0.0044	0.0085	0.0087	0.0087	0.0399	0.1032
Avg. Rank	4.8	4.93	3.40	3.33	2.07	1.80
ECM-NSGA-II ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0
ECM-NSGA-II (p-val)	8.54E-04	1.53E-03	5.37E-03	4.27E-03	-	0.4543
ECM-MOEA/D ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-
ECM-MOEA/D (p-val)	3.05E-04	1.22E-04	5.37E-03	8.54E-04	0.4543	-

compared to ECM-NSGA-II indicate that the former obtains a greater diversity of solutions. This can also be observed from Figure 5.5.

We undertake further comparisons of the Pareto fronts using the Epsilon Indicator (EI) (Lizárraga et al., 2008) which is the minimum factor by which all elements of the control Pareto front must be multiplied to have all its solutions dominated by a candidate Pareto front. Thus, values of EI lower than unity indicate dominance over the control front (for a minimization problem). Table 5.7 lists the EI values with the Pareto fronts of ECM-NSGA-II and ECM-MOEA/D as control. The high values for MOGA and MOGA-SVM show that the Pareto fronts obtained by ECM-NSGA-II as well as ECM-MOEA/D dominate the solutions of MOGA and MOGA-SVM. The low values for ECM-NSGA-II with respect to ECM-MOEA/D

Table 5.6: Comparison of Schott’s Spacing Metric over Synthetic and Real Datasets.

Dataset	MOGA	MOGA-SVM	ECM-NSGA-II	ECM-MOEA/D
Synthetic Datasets				
proximity1	0.0006	0.0057	6.3634	7.3967
proximity2	0.0019	0.0003	6.4726	7.8623
proximity3	0.0302	0.0278	8.0874	10.9807
proximity4	0.0095	0.0094	8.8365	14.2167
proximity5	0.0211	0.0323	6.8797	17.2721
spread1	0.0049	0.0068	10.1438	7.4451
spread2	0.0150	0.0113	8.0367	10.5313
spread3	0.0012	0.0005	7.6440	11.5325
spread4	0.0045	0.0036	8.2167	13.7062
spread5	0.0099	0.0075	8.3366	14.4581
2d-4c-no0	0.0631	0.0357	30.2049	71.8041
2d-4c-no1	0.1985	0.1139	25.8570	68.0962
2d-4c-no2	0.3686	0.4559	20.0541	64.4836
2d-4c-no3	0.0399	0.0383	16.2249	30.4269
2d-4c-no4	0.4166	0.3300	11.1599	25.4085
2d-4c-no5	0.0304	0.0590	24.4067	64.0425
2d-4c-no6	0.1109	0.0605	28.2815	82.7078
2d-4c-no7	0.2497	0.2106	17.7314	37.1530
2d-4c-no8	0.0068	0.0176	20.9853	42.6268
2d-4c-no9	0.1765	0.4639	16.5878	47.5745
10d-4c-no0	0.6852	0.6774	17.6102	18.8409
10d-4c-no1	0.3491	1.3350	11.6804	8.8525
10d-4c-no2	0.2860	0.4526	11.2093	13.4645
10d-4c-no3	0.1105	0.5670	20.8087	17.2844
10d-4c-no4	0.4578	0.7954	18.8652	9.8760
10d-4c-no5	0.3732	0.4289	14.9302	12.1260
10d-4c-no6	0.8254	0.9264	15.3337	16.6048
10d-4c-no7	0.1713	0.2292	23.9735	26.9339
10d-4c-no8	0.2541	0.3944	14.6141	16.7499
10d-4c-no9	0.5031	0.2878	17.1380	23.4287
Real Datasets				
B. scale	3.0929	5.9610	12.4141	10.3693
B. Tissue	0.0005	0.0491	0.0934	1.6077
wdbc	0.0003	0.2422	1.1463	2.1064
banknote	3.9010	5.5631	9.0193	12.8551
echo	0.0283	0.0456	0.8030	1.4726
Ecoli	0.2386	0.3014	9.9088	10.4312
Iris	0.4077	0.2300	1.4639	0.6820
magic	1.8897	9.2838	98.5569	65.4145
seeds	0.4267	0.5103	1.4072	1.1392
sonar	0.5935	0.4166	12.8226	16.0611
ukm	0.5427	0.6283	1.7411	2.9051
wine	0.0515	0.1027	1.9310	2.4930
colon cancer	0.1437	0.2222	46.3518	50.5907
lung cancer	0.3318	0.1213	449.5072	434.2873
prostate cancer	0.2542	0.9114	158.6974	475.1791
Avg. Rank	3.60	3.40	1.78	1.22
ECM-NSGA-II ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	-	\mathcal{H}_1	-
ECM-NSGA-II (p-val)	5.18E-09	5.18E-09	-	3.54E-04
ECM-MOEA/D ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
ECM-MOEA/D (p-val)	5.18E-09	5.18E-09	3.54E-04	-

suggest that the Pareto front of the former generally dominates that of the latter, as can be seen from Figure 5.5 for the *proximity1* dataset.

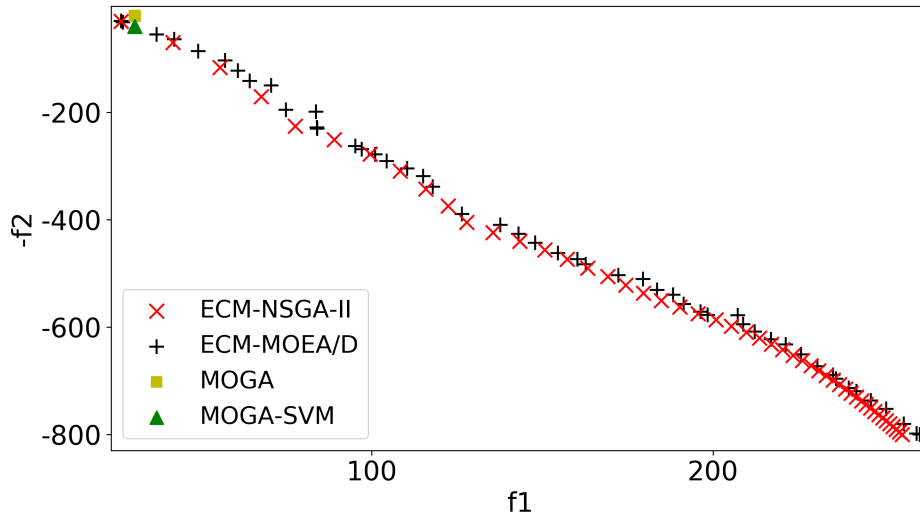


Figure 5.5: Comparison of Pareto fronts on the *proximity1* synthetic dataset

5.3.4 Selection of a Suitable Clustering from the Pareto Set

For unsupervised applications, due to the absence of cluster labels, the best solution may be chosen from the Pareto set found by ECM based on some internal (Wu et al., 2015; Bezdek et al., 2016) or multi-criterion decision making indices (Qu et al., 2016). Alternatively, one can select a suitable trade-off clustering by inspecting the Pareto front.

We present a method to select a suitable clustering from the Pareto fronts identified by ECM-NSGA-II and ECM-MOEA/D. One can observe that starting with the extreme Pareto optimal solution having the lowest value of cluster compactness f_1 , if the points along the Pareto front fall below the line joining the two ends of the front, then there has been a greater increase in entropy f_2 compared to the increase in the value of cluster compactness f_1 . This means that some of the cluster centers have moved slightly closer to each other (evident from a slight increase in cluster compactness f_1) but resulting in a large increase in the entropy f_2 . This is only possible if the clusters in question are truly overlapped because the instances in the region of overlap facilitate a large increase in entropy f_2 . The knee-point from this region of the Pareto front provides an optimal trade-off solution identifying clusters with the appropriate level of overlap.

On the other hand, if the clusterings along the Pareto front were to move above the said line, there would be a larger increase in cluster compactness compared to the increase in entropy. This is only possible if the true clusters are well-separated, and the identified cluster centers have moved closer to each other and away from the true cluster centers. Hence, a deviation above and away from the line joining the end-points indicates that the true clusters are well-separated. In this scenario, the clustering with the minimum value of cluster compactness is the best choice.

Based on these insights, we propose the following method for selecting a clustering from the Pareto front.

Table 5.7: Comparison of Pareto Fronts on Real and Synthetic Datasets using Epsilon Indicator

Dataset	ECM-NSGA-II			ECM-MOEA/D		
	MOGA	MOGA-SVM	ECM-MOEA/D	MOGA	MOGA-SVM	ECM-NSGA-II
Synthetic Datasets						
proximity1	26.6168	26.6180	1.0202	26.6130	26.6142	0.9999
proximity2	11.2888	11.2888	1.0019	11.2883	11.2883	0.9999
proximity3	5.4729	5.4729	1.0288	5.4706	5.4706	0.9996
proximity4	3.2577	3.2577	1.0286	3.2575	3.2575	0.9999
proximity5	2.5073	2.5045	1.0308	2.5072	2.5044	0.9999
spread1	423.3354	439.0631	1.0396	423.0901	438.8086	0.9994
spread2	107.0529	106.6815	1.0304	107.0333	106.6619	0.9998
spread3	37.4213	37.6199	1.0445	37.4083	37.6069	0.9997
spread4	24.1407	24.1058	1.0072	24.1347	24.0998	0.9998
spread5	8.7974	8.8067	1.0049	8.7966	8.8058	0.9999
2d-4c-no0	6.5977	6.5937	1.0464	6.3051	6.3012	0.9999
2d-4c-no1	5.0339	5.0705	1.0154	4.9577	4.9938	0.9999
2d-4c-no2	3.8840	3.4271	1.0212	3.8837	3.3560	0.9999
2d-4c-no3	3.5216	3.5216	1.0020	3.5146	3.5146	0.9999
2d-4c-no4	1.8235	1.8302	1.0348	1.8234	1.8303	0.9999
2d-4c-no5	4.0648	4.0603	1.0001	4.0646	4.0601	0.9999
2d-4c-no6	5.0136	5.0136	1.0026	5.0007	5.0007	0.9999
2d-4c-no7	4.8158	5.3224	1.0027	4.8028	5.3080	0.9999
2d-4c-no8	10.9662	10.8218	1.0070	10.8903	10.8212	0.9999
2d-4c-no9	7.2298	7.1701	1.0149	7.1231	7.0643	0.9999
10d-4c-no0	1.9592	1.9571	1.1234	1.9575	1.9554	0.9991
10d-4c-no1	1.52825	1.5285	1.0540	1.5280	1.5280	0.9997
10d-4c-no2	1.6823	1.6899	1.0068	1.6809	1.6886	0.9992
10d-4c-no3	1.7986	1.7985	1.1495	1.7334	1.7332	0.9993
10d-4c-no4	1.8202	1.8202	1.0941	1.8190	1.8190	0.9993
10d-4c-no5	1.6006	1.6005	1.0466	1.5999	1.5999	0.9996
10d-4c-no6	2.2364	2.2364	1.0561	2.2349	2.2349	0.9993
10d-4c-no7	4.4895	4.4913	1.0346	4.4846	4.4864	0.9989
10d-4c-no8	2.8754	2.8754	1.0443	2.8732	2.8732	0.9992
10d-4c-no9	2.8778	2.8778	1.0614	2.8746	2.8746	0.9989
Real Datasets						
B. scale	1.0130	1.0129	1.0090	1.0080	1.0070	0.9973
B. Tissue	9.9587	9.1640	6.6150	12.0560	11.1250	3.5615
wdbc	144.6750	144.6750	1.0753	134.5460	134.5460	0.9810
banknote	8.7460	8.6840	1.0190	8.7450	8.6820	0.9990
echo	22.6990	22.7130	1.0530	105.7230	105.7820	4.6570
Ecoli	2.6260	3.6970	1.8710	2.5980	3.4520	1.0350
Iris	8.4800	8.2780	1.0080	10.5600	10.3000	1.2440
magic	1.1052	1.1052	1.0000	1.1052	1.1052	1.0047
seeds	25.2700	25.5100	2.3730	17.4700	17.6360	0.6910
sonar	4.1955	4.4084	1.0000	1.0508	1.0079	4.4088
ukm	2.7490	2.7490	0.9596	2.8640	2.8640	1.0590
wine	1.3861	1.3862	1.0145	1.3856	1.3857	0.9997
colon cancer	3.0893	3.0893	0.9999	1.0051	1.0051	3.0737
lung cancer	7.6850	7.6850	1.0000	7.7053	7.7053	1.0026
prostate cancer	2.9077	2.9081	0.9999	2.8927	2.8934	1.0051

- If the first three points⁴ do not lie above the line joining the endpoints, traverse the Pareto front till it touches/ crosses the line joining the endpoints. Choose the clustering corresponding to the point lying at maximum distance from the line within the traversed

⁴Three points are considered to observe the general trend of the front.

region.

- Otherwise, choose the clustering corresponding to the minimum value of cluster compactness f_1 .

For example, let us consider the dataset in Figure 5.1. We observe from the Pareto front in Figure 5.1 that starting from the clustering cI, the Pareto front dips below the line joining point cI and cIV. Therefore, we traverse the Pareto front until it crosses the red line joining the endpoints. Within this region, the clustering cII lies at maximum distance from the red line. Hence using the above method, cII is chosen as the appropriate trade-off clustering. Let us also consider a dataset with three well-separated, equally spaced clusters, which is shown in the first row of Table 5.8 along with the obtained Pareto front. As the Pareto front is observed to rise above the red line joining the extremities, the clustering having the minimum value of cluster compactness f_1 is selected (marked with a red circle). The final cell of the first row of Table 5.8 shows the corresponding clustering. Further demonstrations of the effectiveness of this method over a number of synthetic datasets are shown in the rest of Table 5.8 and Table 5.9.

Table 5.8: The Selection of a Suitable Trade-off Clustering across Different Datasets.

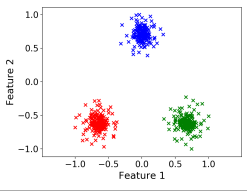
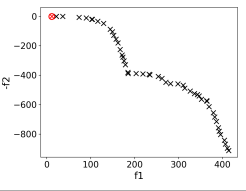
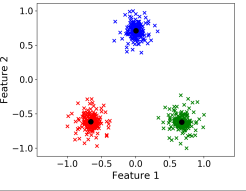
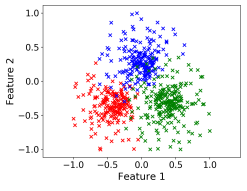
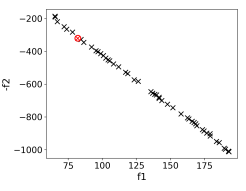
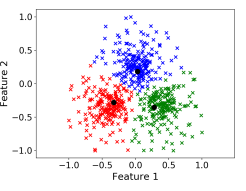
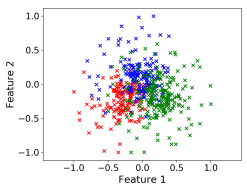
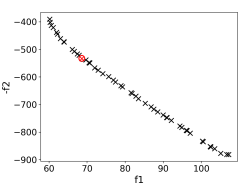
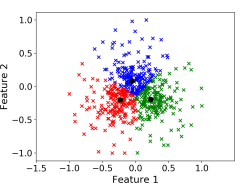
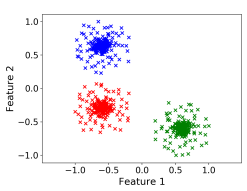
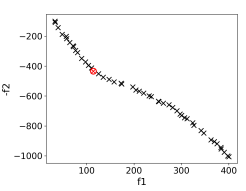
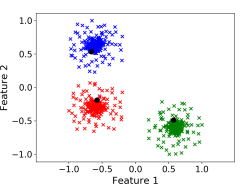
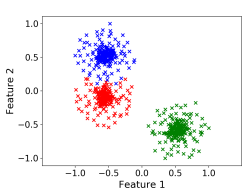
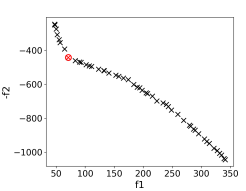
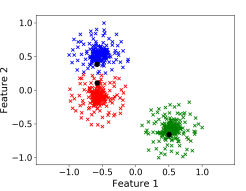
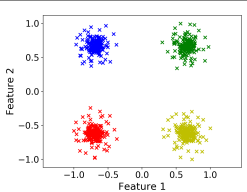
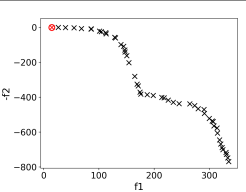
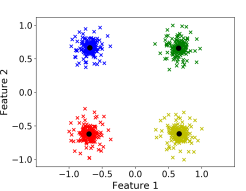
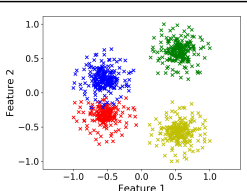
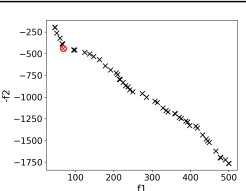
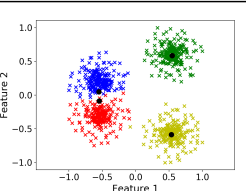
Description	Original Data	Pareto Front	Selected Clustering
3 equally-spaced, well-separated clusters			
3 equally-spaced, slightly overlapped clusters			
3 equally-spaced, highly overlapped clusters			
3 well-separated clusters			
3 clusters, where 2 are slightly overlapped			
4 well-separated clusters			
4 clusters, where 2 are slightly overlapped			

Table 5.9: (Contd. from Table VI) The Selection of a Suitable Trade-off Clustering across Different Datasets.

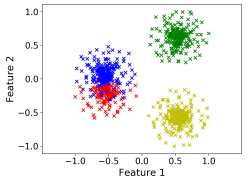
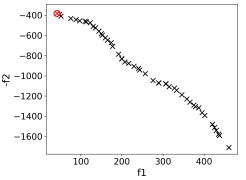
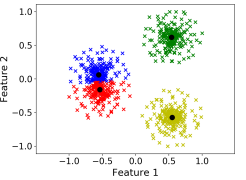
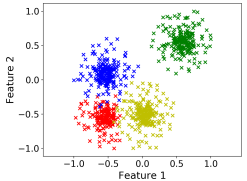
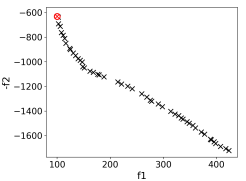
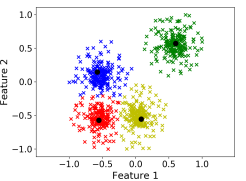
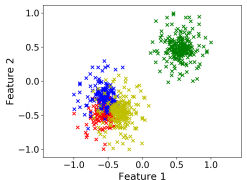
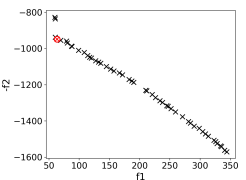
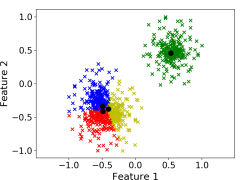
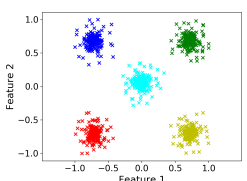
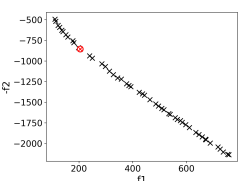
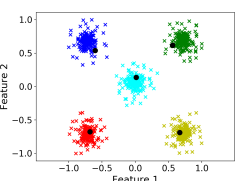
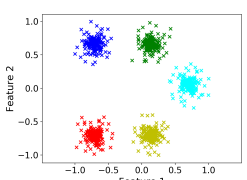
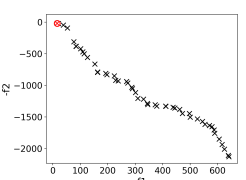
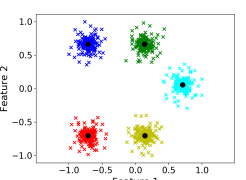
Description	Original Data	Pareto Front	Selected Clustering
4 clusters, where 2 are highly overlapped			
4 clusters, where 3 are slightly overlapped			
4 clusters, where 3 are highly overlapped			
5 well-separated clusters, with 1 in the middle			
5 well-separated clusters, with 1 to the right			

Table 5.10 contains the performance of the proposed method of selecting a clustering from the Pareto front of ECM-NSGA-II and ECM-MOEA/D on the synthetic and real datasets. Here the performance is measured in terms of ARI. For FCM and MEI the mean and standard deviation of the ARI over 50 runs is reported. For MOGA and MOGA-SVM we select the maximum ARI from the Pareto front in each run, and report the mean and standard deviation over 50 runs. For ECM-NSGA-II and ECM-MOEA/D we use the proposed method to select a trade-off clustering in each run. The mean and standard deviation of the ARI of the selected clustering over 50 runs is reported. We observe in Table 5.10 that the top performing methods in order of the average ranks are MOGA, MOGA-SVM, ECM-MOEA/D and ECM-NSGA-II, however the performance of these four methods are statistically comparable. We note that when using the proposed method to select a clustering from the Pareto front, the performance of both ECM-NSGA-II and ECM-MOEA/D is significantly better in comparison to FCM and MEI, and is statistically comparable to MOGA and MOGA-SVM.

Table 5.10: Comparison of ARI over the Artificial and Real Datasets

Dataset	FCM	MEI	MOGA	MOGA-SVM	ECM-NSGA-II	ECM-MOEA/D
proximity1	1.0000	0.8138	1.0000	1.0000	1.0000	1.0000
proximity2	1.0000	0.8884	1.0000	1.0000	1.0000	1.0000
proximity3	1.0000	0.9255	1.0000	1.0000	1.0000	1.0000
proximity4	1.0000	0.8868	1.0000	1.0000	1.0000	1.0000
proximity5	0.8775	0.8775	0.8812	0.8831	0.8864	0.8626
spread1	1.0000	0.9256	1.0000	1.0000	1.0000	1.0000
spread2	1.0000	0.9257	1.0000	1.0000	1.0000	1.0000
spread3	0.9867	0.9199	0.9867	0.9867	0.9867	0.9887
spread4	0.9543	0.8551	0.9543	0.9543	0.9562	0.9640
spread5	0.9479	0.8952	0.9486	0.9479	0.9568	0.9594
2d-4c-no0	0.8834	0.8691	0.8837	0.8834	0.8833	0.8866
2d-4c-no1	0.7997	0.7623	0.7730	0.7748	0.7833	0.8571
2d-4c-no2	0.7766	0.7823	0.8916	0.8909	0.8637	0.7278
2d-4c-no3	0.8333	0.8457	0.9378	0.9378	0.8990	0.9225
2d-4c-no4	0.7297	0.3147	0.7882	0.7880	0.7752	0.7422
2d-4c-no5	0.8557	0.8438	0.9020	0.9046	0.9028	0.8767
2d-4c-no6	0.9547	0.9217	0.9550	0.9552	0.9587	0.9481
2d-4c-no7	0.6424	0.6661	0.6857	0.6839	0.8371	0.8608
2d-4c-no8	0.9000	0.8792	0.9235	0.9234	0.9492	0.9229
2d-4c-no9	0.7814	0.7472	0.8888	0.8890	0.8899	0.8626
10d-4c-no0	0.7967	0.8588	0.7799	0.7891	0.8513	0.8674
10d-4c-no1	0.9797	0.9272	0.9702	0.9795	0.9458	0.9863
10d-4c-no2	0.8844	0.9168	0.8846	0.8934	0.8939	0.9009
10d-4c-no3	0.8875	0.8797	0.8881	0.8878	0.7929	0.8130
10d-4c-no4	0.8121	0.8080	0.8319	0.8142	0.7811	0.8339
10d-4c-no5	0.7178	0.7498	0.7453	0.7399	0.6977	0.7155
10d-4c-no6	0.8727	0.8670	0.8653	0.8590	0.8418	0.8708
10d-4c-no7	0.9940	0.9940	0.9940	0.9940	0.9849	0.9889
10d-4c-no8	0.9603	0.9334	0.9640	0.9639	0.9577	0.9507
10d-4c-no9	0.9577	0.8474	0.9571	0.9526	0.9071	0.9515
B. scale	0.1448	0.1440	0.2886	0.2454	0.1990	0.1447
B. Tissue	0.2885	0.2268	0.2764	0.2763	0.2587	0.2968
wdbc	0.8300	0.8010	0.8300	0.8300	0.8117	0.8104
banknote	0.0452	0.0565	0.0925	0.0976	0.0994	0.1046
echo	0.0854	0.1371	0.0854	0.0854	0.1370	0.1160
Ecoli	0.3684	0.4225	0.4837	0.4621	0.5125	0.4511
Iris	0.7287	0.7652	0.7484	0.7409	0.7548	0.7536
magic	0.0577	0.0275	0.0758	0.0741	0.0764	0.0769
seeds	0.7266	0.4565	0.7266	0.7266	0.6579	0.6799
sonar	0.0064	0.0057	0.0100	0.0147	0.0157	0.0296
ukm	0.1777	0.3227	0.2058	0.2188	0.2320	0.2339
wine	0.8498	0.3786	0.8666	0.8649	0.8421	0.8315
colon	-0.0064	0.0310	0.0122	0.0332	0.0416	0.2995
lung	-0.0003	-0.0093	0.0384	0.0416	0.0388	0.0086
prostate	0.0044	0.0085	0.0087	0.0087	0.0053	0.1032
Avg. Rank	3.56	4.84	2.49	2.53	2.93	2.67
MOGA ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	-	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0
MOGA (p-val)	2.49E-04	2.39E-05	-	0.9064	0.5377	0.7695
MOGA-SVM ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	-	\mathcal{H}_0	\mathcal{H}_0
MOGA-SVM (p-val)	2.78E-04	1.49E-05	0.9064	-	0.3135	0.8342
ECM-NSGA-II ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	-	\mathcal{H}_0
ECM-NSGA-II (p-val)	3.88E-02	4.08E-05	0.5377	0.3135	-	0.3534
ECM-MOEA/D ($\mathcal{H}_0/\mathcal{H}_1$)	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_0	\mathcal{H}_0	\mathcal{H}_0	-
ECM-MOEA/D (p-val)	3.78E-03	1.42E-05	0.7695	0.8342	0.3534	-

5.4 Discussions

We propose a fuzzy center-based clustering method by ECM using the MOO methods NSGA-II and MOEA/D, to produce fuzzy clusterings at different levels of fuzziness. The proposed methods are able to identify clusters with different levels of overlap, with ECM-NSGA-II producing slightly better results on lower dimensional datasets. On the other hand the MOEA/D variant shows better results on higher dimension datasets, and is generally observed to produce more uniformly spaced clusterings along Pareto fronts. While the results of ECM-NSGA-II and ECM-MOEA/D are statistically comparable, we recommend the use of ECM-NSGA-II for lower dimensional datasets and the use of ECM-MOEA/D on higher dimensional datasets since the methods were observed to generally have a slight edge in performance in these cases. Additionally, we present a method to select a suitable trade-off clustering from the Pareto front. Future investigations can be towards identifying an appropriate number of clusters, incorporating different distance metrics such as in multiple kernel clustering (Chen et al., 2011; Liu et al., 2017b), or towards different methods such as fuzzy possibilistic clustering (Tsai et al., 2012; Saha and Das, 2018).

Chapter 6

Transfer Clustering using Multiple Kernel Metrics Learned under Multi-Instance Weak Supervision

Summary

Multiple kernel clustering methods have been quite successful recently especially concerning the multi-view clustering of complex datasets. These methods simultaneously learn a multiple kernel metric while clustering in an unsupervised setting. With the motivation that some minimal supervision can potentially increase their effectiveness, we propose a Multiple Kernel Transfer Clustering (MKTC) method that can be described in terms of two tasks: a source task, where the multiple kernel metric is learned, and a target task where the multiple kernel metric is transferred to partition a dataset. In the source task, we create a weakly supervised multi-instance subset of the dataset, where a set of data instances are together provided some labels. We put forth a Multiple Kernel Multi-Instance k-Means (MKMIKM) method to simultaneously cluster the multi-instance subset while also learning a multiple kernel metric under weak supervision. In the target task, MKTC transfers the multiple kernel metric learned by MKMIKM to perform unsupervised single-instance clustering of the entire dataset in a single step. The advantage of using a multi-instance setup for the source task is that it requires reduced labeling effort to guide the learning of the multiple kernel metric. Our formulations lead to a significantly lower computational cost in comparison to the state-of-the-art multiple kernel clustering algorithms, making them more applicable to larger datasets. Experiments over benchmark computer vision datasets suggest that MKTC can achieve significant improvements in clustering performance in comparison to the state-of-the-art unsupervised multiple-kernel clustering methods and other transfer clustering methods.

6.1 Introduction

Among the various factors affecting center-based clustering, two of the factors that were discussed in Section 1.2 of Chapter 1 were the features of the data instances considered and the distance metric used. As clustering methods typically work in an unsupervised setting, the features and the distance metric are generally predetermined. Some flexibility

in the choice of features can be provided through feature weighting (Huang et al., 2005) and subspace clustering approaches (Elhamifar and Vidal, 2013; Soltanolkotabi et al., 2014). Metric learning approaches allow learning a distance metric that benefits cluster recognition (Xing et al., 2003; Bilenko et al., 2004; Basu et al., 2010). Among metric learning approaches, multiple kernel clustering methods (Zhao et al., 2009a; Lu et al., 2014; Du et al., 2015; Liu et al., 2016; Li et al., 2016; Wang et al., 2017; Liu et al., 2017b) have the advantage of learning from a wide range of possible metrics. This is achieved by learning a combination of several predetermined kernel metrics that best fits a dataset. This advantage has been instrumental in improving the performance of recent unsupervised multiple kernel clustering algorithms (Zhu et al., 2018; Liu et al., 2019; Liu et al., 2019; Liu et al., 2020b; Zhou et al., 2020; Liu et al., 2020a,c; Han et al., 2018a,b; Ren and Sun, 2020). Some additional supervision may potentially improve clustering performances, by increasing the bias of the clustering model to learn the relevant features or a suitable distance metric that better fits the data at hand (Law et al., 2017; Ji et al., 2019).

Ideally, under full supervision, all the data points are provided with accurate labels. Hence, the most informed selection of features and distance metrics is feasible. However in several domains of applications, obtaining labels for all data points from experts can be too expensive, making it infeasible for large datasets. Broadly there are four categories of approaches to deal with the lack of full supervision (Ratner et al., 2019 (accessed July 16, 2021); (Ratner et al., 2019, 2020). *Active Learning* (Xiong et al., 2017; Basu et al., 2004a; Grira et al., 2008; Xiong et al., 2013) aims to estimate the best subset of data instances that experts can assign new labels to. *Semi-supervised Learning* (Basu et al., 2006; Bair, 2013; Kulis et al., 2009; Yu et al., 2015; Soares et al., 2017) finds the most appropriate way to utilize a usually larger set of unlabeled data along with a smaller labeled dataset to improve prediction. *Transfer Learning* (Pan and Yang, 2010; Deng et al., 2016; Jiang and Chung, 2012; Han et al., 2019) involves the use of models that have already been trained on different tasks in usually a similar domain of application. *Weak Supervision* (Zhou, 2017; Ratner et al., 2020; Xu et al., 2015; Durand et al., 2017; Zhou et al., 2018) focuses on acquiring easier to obtain but potentially noisier labels, often at a higher level of abstraction than that of instance-level labeling. In the field of clustering, Law et al. (2017) was motivated by the weak supervision approach of *Multi-Instance Learning* (Jin et al., 2009; Dietterich et al., 1997) that forms sets of data instances called bags, and assigns a vector of class labels to each bag, indicating whether a class has an instance in that bag. This assignment of labels to a higher level of abstraction (i.e., to the bags) instead of to the individual data instances requires less labeling effort from experts. Based on this approach, Law *et. al* proposed a weakly supervised multi-instance clustering model that learns a Mahalanobis distance while clustering the dataset (Law et al., 2017).

We are motivated by the notion of including some degree of supervision to improve the identification of clusters in a dataset, while also requiring the associated labeling effort to be low, as was discussed in Section 1.4 of Chapter 1. Therefore we propose a method called Multiple Kernel Transfer Clustering (MKTC) which aims to cluster a dataset using a multiple kernel metric learned under weak supervision. The clustering method is divided into a *source task* and a *target task*, which is visually summarised in Figure 6.1. The source task learns a multiple kernel metric under weak supervision. This is accomplished by constructing a multi-instance subset of the dataset, where multiple data instances together form a bag. To each bag, weakly supervised labels are assigned by an expert in the form of a vector of possible

classes that may have instances in the bag. We propose a new Multiple Kernel Multi-Instance k -Means (MKMIKM) objective to learn a multiple kernel metric while simultaneously clustering the weakly supervised multi-instance subset. The MKMIKM objective is optimized using an alternating-optimization (AO) algorithm that has computation cost linear in the size of the dataset, in comparison to the state-of-the-art multiple kernel methods that have a higher cost that is quadratic in the size of the dataset. In the target task of MKTC, we transfer in the multiple kernel metric learnt in the source task, and perform unsupervised single-instance clustering of the original dataset in a single step.

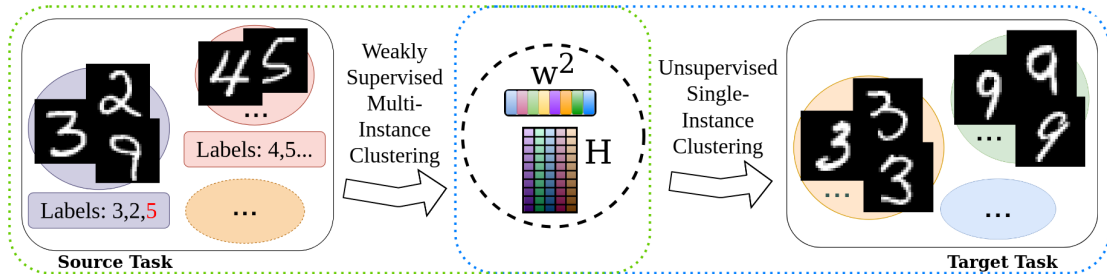


Figure 6.1: The proposed Multiple Kernel Transfer Clustering (MKTC) method, described in terms of a source task and a target task. In the source task, a multi-instance subset of the dataset is constructed, where multiple data instances together form a bag to which weakly supervised labels are assigned in the form of a (possibly noisy) vector of classes that may have instances in the bag. The source task subset is clustered by our proposed Multiple Kernel Multi-Instance k -Means (MKMIKM), which simultaneously learns a multiple kernel metric parameterized by w . In the target task of MKTC, w is transferred in from the source task along with the cluster membership matrix H to perform unsupervised single-instance clustering of the entire dataset in a single step.

To the best of our knowledge, transfer learning has not been previously used in the context of center-based clustering, where model parameters are transferred between vastly different source and target tasks, such as between a weakly supervised multi-instance source task and an unsupervised single-instance target task for the proposed MKTC. Hence we also define a formal notion of transfer learning for center-based clustering approaches, which serves as a framework that we follow to define our proposed method of MKTC. Thus the contributions of our work presented in this chapter are as follows:

- We formalize the notion of transfer learning for center-based clustering in Section 6.2.
- We propose a novel method called MKTC based on this framework, which is divided into a source task where a multiple kernel metric is learned, and a target task where the metric is transferred to cluster the dataset.
- In the source task, the multiple kernel metric is learned in a multi-instance weakly supervised setup, which allows for reduced labeling effort from experts in comparison to a fully supervised setup.
- The multiple kernel metric is learned by our proposed MKMIKM AO algorithm in $O(mkn^{(s)})$ time, in comparison to the $O(mkn^{(s)^2})$ time required by the state-of-the-art

multiple kernel methods, where m is the number of kernels, k is the number of clusters, $n^{(s)}$ is the size of the source task dataset.

- The target task of our proposed MKTC transfers in the multiple kernel metric learned in the source task to bias the method and perform single-instance unsupervised clustering of the entire dataset in a single step.
- The overall computation cost of MKTC is $O(mkn^{(s)}n^{(t)})$, where $n^{(t)}$ is the size of the target task dataset and $n^{(s)} < n^{(t)}$. This is an important reduction in cost from the $O(mkn^{(t)^2})$ cost required by state-of-the-art multiple kernel clustering methods to cluster the target task dataset, making the proposed method better suited for clustering large datasets.

The rest of the chapter is thus organized in the following way. In Section 6.2, the formal notion of transfer learning for center-based clustering is defined. We discuss the related literature on multi-instance clustering under weak supervision and multiple kernel clustering in Section 6.3. In Section 6.4 we discuss the proposed approach of MKTC, in terms of the multi-instance weakly supervised approach to learn the multiple kernel distance metric in Section 6.4.1, and following which the proposed single-instance unsupervised clustering approach is discussed in Section 6.4.2. The experiments studied the efficacy of the proposed methods are discussed in Section 6.5, and finally our concluding remarks on this chapter are provided in Section 6.6.

6.2 Transfer Learning for Center-Based Clustering

Transfer Learning aims to bias a statistical model by learning model parameters in a source task and using some or all of the learned parameters to improve learning in a different target task (Pan and Yang, 2010; Zhuang et al., 2021). Transfer Learning has been observed to be effective when the source and target tasks are in similar domains of application, with additional experience or expertise often being required to perform well in very different domains (Rosenstein et al., 2005; Ying et al., 2018). For a general center-based clustering model, a framework for transfer learning can be defined in terms of first learning model parameters from a source task (superscripted by $^{(s)}$):

$$\min \sum_{i=1}^n \sum_{j=1}^k H_{ij}^{(s)} d_{f^{(s)}}^{(s)}(f^{(s)}(\mathbf{x}_i^{(s)}), f^{(s)}(\mathbf{v}_j^{(s)})).$$

Here the source task dataset $X^{(s)} = \{\mathbf{x}_1^{(s)}, \dots, \mathbf{x}_n^{(s)}\}$, $\mathbf{x}_i^{(s)} \in \mathbb{R}^d$, is clustered by estimating the cluster centers $V^{(s)} = \{\mathbf{v}_1^{(s)}, \dots, \mathbf{v}_k^{(s)}\}$, $\mathbf{v}_j^{(s)} \in \mathbb{R}^d$, and the cluster membership matrix $H^{(s)} \in [0, 1]^{n \times k}$. In addition, transformations of the data space $f^{(s)}(\cdot)$ can be learned if feature selection or extraction is considered (e.g. sparse clustering, subspace clustering, etc.), and the distance metric $d_{f^{(s)}}^{(s)}$ defined in the transformed space can also be learned. From this framework, $f^{(s)}$ or $d_{f^{(s)}}^{(s)}$ can be learned from a source task under some degree of supervision (Bengio, 2012; Pan et al., 2008), and transferred to the target task (superscripted by $^{(t)}$)

involving dataset $X^{(t)}$ where the now biased learner can estimate $V^{(t)}$ and $H^{(t)}$:

$$\min \sum_{i=1}^n \sum_{j=1}^k H_{ij}^{(t)} d_{f^{(s)}}^{(s)}(f^{(s)}(\mathbf{x}_i^{(t)}), f^{(s)}(\mathbf{v}_j^{(t)})).$$

In this chapter, we propose a Multiple Kernel Transfer Clustering method which, following this framework, has a source task and a target task. The source task involves estimating a multiple kernel metric on a multi-instance weakly supervised subset of a dataset. The multiple kernel metric learned from this source task is then transferred to the target task of single-instance unsupervised clustering of the entire dataset. The proposed source and target tasks are defined in Sections 6.4.1 and 6.4.2. Although the clustering tasks are different, both tasks lie in the same domain, thereby leading us to expect improved clustering performance.

In the related area of semi-supervised clustering, improvements in clustering were achieved by introducing some degree of supervision when estimating model parameters or clustering constraints on a fully supervised subset of a dataset (Basu et al., 2004b; Bai et al., 2020). We differentiate semi-supervised approaches from the defined transfer clustering framework in the following way: In semi-supervised learning, the source and target clustering tasks are similar in nature and generally operate on the same dataset, whereas in transfer learning we can consider vastly different source and target tasks. This allows us to propose our method involving a multi-instance weakly supervised clustering source task, and a single-instance unsupervised clustering target task.

6.3 Related Works

In this section, we discuss the areas of multi-instance weakly supervised clustering, followed by a discussion on multiple kernel clustering. These two areas motivate our proposed method of MKTC, described in Section 6.4.

6.3.1 Multi-Instance Clustering under Weak Supervision

In Multi-Instance learning, a dataset $X \in \mathbb{R}^{n \times d}$ is formed from sets of data instances $X = [X_1^T, \dots, X_b^T]^T$, where each set is called a bag and can contain variable number of data instances $X_i \in \mathbb{R}^{n_i \times d}$, $\sum_{i=1}^b n_i = n$. Additionally a bag label matrix $Y \in \{0, 1\}^{b \times k}$ is formed, where each row $\mathbf{y}_i \in \{0, 1\}^k$ informs which of the k categories have instances in bag i . The labeling effort can be greatly reduced by asking an expert to provide labels under the following *relaxed conditions*: (i) at least one class needs to be labeled that has an instance in that bag, and (ii) it is not necessary for the class of each data instance to be labeled by an expert, nor is it necessary for all classes that have instances in a bag to be labeled.

Law et al. (2017) proposed an objective of a Multi-Instance k -Means clustering model that estimates cluster centers $V = [\mathbf{v}_1^T, \dots, \mathbf{v}_k^T]^T \in \mathbb{R}^{k \times d}$ representing each of the k categories, and a cluster assignment matrix $H = [H_1^T, \dots, H_b^T]^T \in \{0, 1\}^{n \times k}$, $H_i \in \{0, 1\}^{n_i \times k}$ with the aim of predicting the correct category of each data instance. The following two constraints were imposed on H to take into account the relaxed conditions of assigning labels to each bag. The first constraint restricted the assignment of an instance to at most one category, stated formally as $H_i \mathbf{1} \preceq \mathbf{1}$. The second constraint imposed that for the i -th bag, at most one data instance can be assigned to each of the candidate categories for bag i . This was equivalent

to making $\min\{n_i, \mathbf{y}_i^T \mathbf{1}\}$ cluster assignments for each bag, while maintaining $H_i^T \mathbf{1} \preceq \mathbf{y}_i$. Formally, the second constraint was stated as: $\{H_i^T \mathbf{1} \preceq \mathbf{y}_i, \mathbf{1}^T H_i \mathbf{1} = \min\{n_i, \mathbf{y}_i^T \mathbf{1}\}\}$. Thus H was constrained to the following consistency set Q^{\cup} :

$$Q^{\cup} = \{H = [H_1^T, \dots, H_b^T]^T : \forall i, H_i \in \mathcal{U}_i\},$$

$$\text{where, } \mathcal{U}_i = \{H_i \in \{0, 1\}^{n_i \times k} : H_i \mathbf{1} \preceq \mathbf{1},$$

$$H_i^T \mathbf{1} \preceq \mathbf{y}_i, \mathbf{1}^T H_i \mathbf{1} = \min\{n_i, \mathbf{y}_i^T \mathbf{1}\}\}.$$

Based on these definitions, the objective of the Multi-Instance k -Means was defined as,

$$\min_{H \in Q^{\cup}, V \in \mathbb{R}^{k \times d}} \sum_{i=1}^n \sum_{j=1}^k H_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|^2. \quad (6.1)$$

Law et al. (2017) showed that a local minima for this objective could be obtained by alternate optimization over H and V . This multi-instance clustering setup where low effort supervision can be provided motivates us to form the source task of MKTC where a multiple kernel metric is learnt under weak supervision, which is described in Section 6.4.1.

6.3.2 Multiple Kernel Clustering

To cluster a dataset $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$, $\mathbf{x}_i \in \mathbb{R}^d$, early works on multiple kernel clustering (Zhao et al., 2009a; Chen et al., 2011; Huang et al., 2012a) were based on the formulation of a multiple kernel k -Means *with centers in the kernel space* (Schölkopf et al., 1998; Girolami, 2002; Dhillon et al., 2004; Jha et al., 2020),

$$\min_{H, \mathbf{w}, V} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^n H_{ij} w_l^\beta \|\phi_l(\mathbf{x}_i) - \mathbf{v}_j\|^2, \quad (6.2)$$

subject to (s.t.) the constraint $\sum_{l=1}^m w_l = 1$, and a suitable choice of β . Here $\phi_l(\cdot)$ are non-linear functions that map the data instances to a higher dimension Hilbert space. Expansion of the squared norm allows the substitution of dot products $\phi_l^T(x_i)\phi_l(x_j)$ with kernel functions $K_l(x_i, x_j)$. The multiple kernel metric between two instances x_i and x_j is specified by the weighted combination of kernels $\sum_{l=1}^m w_l^\beta K_l(x_i, x_j)$. In Liu et al. (2016); Li et al. (2016), a new way of performing multiple kernel clustering was introduced that yielded significant improvements in clustering performance, where the objective function was reformulated as a trace maximization problem involving n^2 -sized kernel matrices containing the multiple kernel metric computed between all pairs of data instances. Subsequent developments based on the trace maximization objective introduced ways to relax the search space of the combined kernel matrix (Liu et al., 2017b; Liu et al., 2020a). Different ways to improve on the base kernel matrices were investigated, through outlier correction (Wang et al., 2017) or by considering neighbourhoods of kernel similarities (Zhou et al., 2020). Due to the large size of the kernel matrices, it may be possible for several entries to be absent or corrupted, prompting a line of research on working with incomplete kernels (Zhu et al., 2018; Liu et al., 2019; Liu et al., 2019; Liu et al., 2020b). However all methods following this line of research either directly optimize the objective (6.2), or its reformulated trace maximization objective, leading to high

computational costs of $O(mkn^2)$ or higher, due to having to deal with m number of n^2 -sized kernel matrices. This limits the applicability of these methods to datasets with lower number of data instances.

6.4 Methodology

In this section, our proposed method of Multiple Kernel Transfer Clustering (MKTC) is described in terms of two tasks: a source task and a target task. For the source task, we propose an alternating optimization (AO) algorithm of Multiple Kernel Multi-Instance k -Means (MKMIKM) to learn a multiple kernel metric while clustering a multi-instance subset of a dataset under weak supervision, described in 6.4.1. The target task uses this learned multiple kernel metric to perform clustering of the entire dataset in a single step, described in 6.4.2.

6.4.1 Multiple Kernel Multi-Instance k -Means clustering

For a dataset X , in our source task we wish to learn a multiple kernel metric in a weakly supervised multi-instance setup, as shown in Figure 6.1. Following our defined framework of transfer learning for center-based clustering as described in Section 6.2 and motivated by the setup of Law et al. (2017), we construct a multi-instance subset of X for our source task $X^{(s)} \subset X$, where $X^{(s)}$ consists of b bags $X^{(s)} = [X_1^{(s)T}, \dots, X_b^{(s)T}]^T \in \mathbb{R}^{n^{(s)} \times d}$ and each bag can contain variable number of data instances $X_i^{(s)} \in \mathbb{R}^{n_i \times d}$, $\sum_{i=1}^b n_i = n^{(s)}$. A bag label matrix $Y^{(s)} \in \{0, 1\}^{b \times k}$ is also provided, where each row $\mathbf{y}_i^{(s)}$ indicates which of the k categories have instances in bag i . In our source task, we aim to cluster this dataset $X^{(s)}$, which involves estimating cluster centers $V^{(s)} = [\mathbf{v}_1^{(s)T}, \dots, \mathbf{v}_k^{(s)T}]^T \in \mathbb{R}^{k \times d}$, and cluster assignment matrix $H^{(s)} = [H_1^{(s)T}, \dots, H_b^{(s)T}]^T \in \{0, 1\}^{n^{(s)} \times k}$, $H_i \in \{0, 1\}^{n_i \times k}$. The same constraints are imposed on $H^{(s)}$ restricting the assignment of an instance to at most one category, and for each bag restricting assignments to at most one data instance for each candidate cluster. Thus $H^{(s)}$ is constrained to the following consistency set $Q^{U^{(s)}}$:

$$\begin{aligned} Q^{U^{(s)}} &= \{H^{(s)} = [H_1^{(s)T}, \dots, H_b^{(s)T}]^T : \forall i, H_i^{(s)} \in U^{(s)}_i\}, \\ &\text{where, } U^{(s)}_i = \{H_i^{(s)} \in \{0, 1\}^{n_i \times k} : H_i^{(s)} \mathbf{1} \preceq \mathbf{1}, \\ &\quad H_i^{(s)T} \mathbf{1} \preceq \mathbf{y}_i^{(s)}, \mathbf{1}^T H_i^{(s)} \mathbf{1} = \min\{n_i, \mathbf{y}_i^{(s)T} \mathbf{1}\}\} \end{aligned}$$

We wish to learn a multiple kernel metric in this multi-instance weak supervision setup. The literature on multiple kernel clustering has focused on objective functions based on multiple kernel k -Means with centers in the kernel space (6.2), however this clustering setup is not ideal for larger datasets. For datasets of size $n^{(s)}$, the computation cost incurred is $O(mkn^{(s)2})$, which is too expensive for larger computer vision datasets: e.g., the MNIST dataset with $N = 70,000$ data instances, construction of $m = 7$ 32-bit kernels would require ≈ 1022 gigabits of memory.

To develop a lower computation method, one alternative is to adapt kernel clustering with *centers in the feature space* (Ferreira and de A.T. de Carvalho, 2014; Gupta and Das, 2017)

to a multiple kernel setup,

$$\min_{H^{(s)}, \mathbf{w}, V^{(s)}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} w_l^\beta \{ \|\phi_l(\mathbf{x}_i^{(s)}) - \phi_l(\mathbf{v}_j^{(s)})\|^2 \}.$$

s.t. $\sum_{l=1}^m w_l^{(s)} = 1$. Single kernel clustering methods with centers in the feature space can have $O(kn^{(s)})$ computation cost instead of $O(n^{(s)^2})$. However they have been largely abandoned in the research on multiple kernel clustering. Our motivation is to form a multiple kernel multi-instance clustering method with $O(mkn^{(s)})$ computation cost, making them applicable to larger datasets. We thus propose the following Multiple Kernel Multi-Instance k -Means (MKMIKM) problem with *centers in the feature space*,

$$\begin{aligned} \min_{H^{(s)} \in Q^{U^{(s)}}, \mathbf{w}^{(s)}, V^{(s)}} & \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} w_l^{(s)^2} \{ \alpha \\ & + \|\phi_l(\mathbf{x}_i^{(s)}) - \phi_l(\mathbf{v}_j^{(s)})\|^2 \}, \quad \text{s.t. } \mathbf{w}^{(s)T} \mathbf{1} = 1. \end{aligned} \quad (6.3)$$

The additive α ensures larger gradients of the objective. Expanding the squared norm in (6.3) and replacing all inner products $\phi_l(\mathbf{x}_i^{(s)})^T \cdot \phi_l(\mathbf{v}_j^{(s)})$ with corresponding kernel functions $K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})$ leads to the following MKMIKM objective using Gaussian kernels,

$$\begin{aligned} \min_{H \in Q^{U^{(s)}}, \mathbf{w}^{(s)}, V^{(s)}} & \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} w_l^{(s)^2} \{ \alpha + K_l(\mathbf{x}_i^{(s)}, \mathbf{x}_i^{(s)}) \\ & + K_l(\mathbf{v}_j^{(s)}, \mathbf{v}_j^{(s)}) - 2K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}) \}, \quad \text{s.t. } \mathbf{w}^{(s)T} \mathbf{1} = 1. \end{aligned} \quad (6.4)$$

When working with multiple kernel methods, several kernel functions are available for the choice of each K_l (Genton, 2002; Hofmann et al., 2008). The optimization of $\mathbf{w}^{(s)}$ and $H^{(s)}$ can be carried out efficiently regardless of the choice of kernel functions, however the cost of optimizing $V^{(s)}$ depends on which kernel functions are used. If kernel functions are selected so that the derivative of the Lagrangian of (6.4) with respect to $V^{(s)}$ yields a closed-form solution, then an overall $O(mkn^{(s)})$ algorithm can be derived. In the absence of closed-form solutions, gradient descent based approaches can be used to optimize $V^{(s)}$, which increases the computation cost to $O(mkn^{(s)}T_{\max})$, where T_{\max} is the maximum number of iterations allowed for the descent algorithm. To keep the computation cost as low as possible, Gaussian kernels can be used in objective (6.4), which yield closed form solutions when optimizing $V^{(s)}$. Gaussian kernels have the form $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$. When $\mathbf{x}_i = \mathbf{x}_j$, $K(\mathbf{x}_i, \mathbf{x}_i) = 1$. Using Gaussian kernels and setting α to 2 reduces the objective (6.4) to the following,

$$\min_{H^{(s)} \in Q^{U^{(s)}}, \mathbf{w}^{(s)}, V^{(s)}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} w_l^{(s)^2} \{ 2 - K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}) \}, \quad (6.5)$$

s.t. $\mathbf{w}^{(s)T} \mathbf{1} = 1$. The derivative of the Lagrangian of (6.5) with respect to $\mathbf{v}_j^{(s)}$ yields the

following update expression,

$$\mathbf{v}_j^{(s)} = \frac{\sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} \zeta_{ij} \mathbf{x}_i^{(s)}}{\sum_{i=1}^{n^{(s)}} H_{ij}^{(s)} \zeta_{ij}}, \quad (6.6)$$

where $\zeta_{ij} = \sum_{l=1}^m w_l^{(s)2} \sigma_l^{-2} K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})$. Similarly, the derivative of the Lagrangian of (6.5) with respect to $w_l^{(s)}$ yields the following update expression,

$$w_l^{(s)} = \frac{[\sum_{i=1}^{n^{(s)}} \sum_{j=1}^k H_{ij}^{(s)} \{2 - K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})\}]^{-1}}{\sum_{l'=1}^m [\sum_{i=1}^{n^{(s)}} \sum_{j=1}^k H_{ij}^{(s)} \{2 - K_{l'}(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})\}]^{-1}}. \quad (6.7)$$

To obtain an optimal $H^{(s)}$, we hold $V^{(s)}$ and $\mathbf{w}^{(s)}$ constant and decompose (6.5) into b objectives for each bag,

$$\min_{H_i^{(s)} \in \mathcal{U}^{(s)}} \sum_{l=1}^m \sum_{j=1}^k \mathbf{H}_{ij}^{(s)} w_l^{(s)2} \{2 - K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})\}. \quad (6.8)$$

The optimal solution of objective (6.8) is obtained by the Hungarian algorithm (Kuhn, 1955). Let $p_i = \min\{n_i, \mathbf{y}_i^T \mathbf{1}\}$, and $q_i = \max\{n_i, \mathbf{y}_i^T \mathbf{1}\}$. The Hungarian algorithm constructs a table of size $p_i \times q_i$ of scores, where each data instance is assigned a score corresponding to each cluster, based on the objective (6.8). For each cluster, a data instance is assigned to it based on the lowest score from the table, such that no two clusters are assigned the same data instance. This can be done in $O(p_i^2 q_i)$ time (Bourgeois and Lassalle, 1971). As bag sizes are never considered to be too large ($\forall i n_i < 15$) (Law et al., 2017), this is efficient to compute.

Based on the above discussions, Algorithm 6 outlines the AO algorithm to optimize objective function (6.5). Algorithm 6 returns the kernel weights $\mathbf{w}^{(s)}$ which along with the base kernel functions, forms the multiple kernel metric.

Algorithm 6 Multiple Kernel Multi-Instance k -Means (MKMIKM)

Input: Data bags $X^{(s)}$, bag labels $Y^{(s)}$.

Output: Kernel weights $\mathbf{w}^{(s)}$, cluster assignments $H^{(s)}$.

- 1: Initialize random $V^{(s)}$ and $H^{(s)}$, set each $w_l^{(s)} := \frac{1}{m}$.
 - 2: Compute initial $K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}) \forall l, \mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}$.
 - 3: **repeat**
 - 4: Update $H^{(s)}$ by solving eqn. (6.8).
 - 5: Update $V^{(s)}$ using eqn. (6.6).
 - 6: Recompute $K_l(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}) \forall t, \mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)}$.
 - 7: Update $\mathbf{w}^{(s)}$ using eqn. (6.7).
 - 8: **until** convergence
-

On Complexity and Convergence: In Algorithm 6, computing $H^{(s)}$ requires running the $O(p_i^2 q_i)$ Hungarian algorithm (Bourgeois and Lassalle, 1971) for all b bags, resulting in a

total of $O(bp_i^2q_i)$ time. Updating V from (6.6), computing all $K_t(\mathbf{x}_i^{(s)}, \mathbf{v}_j^{(s)})$, and updating $\mathbf{w}^{(s)}$ from (6.7), each require $O(mkn^{(s)})$ time. This leads to a total computation cost of $O(mkn^{(s)} + bp_i^2q_i)$. A reasonable constraint can be imposed on the bag sizes so that $p_i < \sqrt{k}$ and $q_i \approx n^{(s)}/b$, leading to an overall computational cost of $O(mkn^{(s)})$.

Empirical convergences across different datasets are shown in Section 6.5.6. Numerical convergence is monitored by setting a maximum number of iterations, while also checking for changes in $H^{(s)}$ across successive iterations.

6.4.2 Multiple Kernel Transfer Clustering

As discussed in the previous section, for the source task of MKTC we proposed MKMIKM to learn a multiple kernel metric from a weakly supervised multi-instance subset $X^{(s)} \subset X$ of a dataset X . In the target task of MKTC, we intend to perform unsupervised single-instance clustering of our original dataset $X^{(t)} = X$ of size $n^{(t)}$, by transferring in the multiple kernel metric learned by MKMIKM to bias MKTC to better fit the dataset, as shown in Figure 6.1. Whereas the source task is resolved by iterating over an AO algorithm, the target task is designed to require a single step of the computation.

Before we define the objective function for our target task, let us first consider a multi-instance weakly supervised multiple kernel clustering objective based on (6.2),

$$\min_{H \in Q^{\mathcal{G}, \mathbf{w}, V}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^n H_{ij} w_l \|\phi_l(\mathbf{x}_i) - \mathbf{v}_j\|, \text{ s.t. } \mathbf{w}^T \mathbf{1} = 1.$$

Expanding the norm yields the following objective,

$$\begin{aligned} \min_{H \in Q^{\mathcal{G}, \mathbf{w}, V}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^n H_{ij} w_l \{ \phi_l(\mathbf{x}_i) \cdot \phi_l(\mathbf{x}_i) \\ - 2\phi_l(\mathbf{x}_i) \cdot \mathbf{v}_j + \mathbf{v}_j \cdot \mathbf{v}_j \}, \text{ s.t. }, \mathbf{w}^T \mathbf{1} = 1, H \mathbf{1} = \mathbf{1}. \end{aligned}$$

The derivative of this objective with respect to \mathbf{v}_j yields,

$$\mathbf{v}_j = \frac{\sum_{l=1}^m \sum_{i=1}^n H_{ij} w_l \phi_l(\mathbf{x}_i)}{\sum_{l=1}^m \sum_{i=1}^n H_{ij} w_l}. \quad (6.9)$$

Now we propose the objective function for the target task of multiple kernel single-instance unsupervised clustering,

$$\min_{U^{(t)}, \omega^{(t)}, V^{(t)}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(t)}} \mu_{ij}^{(t)} \omega_l^{(t)} \|\phi_l(\mathbf{x}_i^{(t)}) - \mathbf{v}_j\|^2,$$

s.t., $\omega^{(t)T} \mathbf{1} = 1, U^{(t)} \in \{0, 1\}^{n^{(t)} \times k}, U^{(t)} \mathbf{1} = \mathbf{1}$. Here $\omega^{(t)} \in [0, 1]^m$ is the parameter of the multiple kernel metric, and $U^{(t)}$ is the matrix of cluster memberships $\mu_{ij}^{(t)}$ between data points $\mathbf{x}_i^{(t)}$ and the j -th cluster. Using eqn. (6.9) we transfer in variables from our source

task, forming the following objective,

$$\min_{U^{(t)}, \omega^{(t)}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(t)}} \mu_{ij}^{(t)} \omega_l^{(t)} \left\{ \left\| \phi_l(\mathbf{x}_i^{(t)}) - \frac{\sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_{l'}^{(s)} \phi_{l'}(\mathbf{x}_{i'}^{(s)})}{\sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_{l'}^{(s)}} \right\|^2 \right\},$$

s.t., $\omega^{(t)T} \mathbf{1} = 1, U^{(t)} \mathbf{1} = \mathbf{1}$. Expanding the squared norm and dropping the first term, we obtain the following objective,

$$\begin{aligned} & \min_{U^{(t)}, \omega^{(t)}} \sum_{l=1}^m \sum_{j=1}^k \sum_{i=1}^{n^{(t)}} \mu_{ij}^{(t)} \omega_l^{(t)} \left\{ \right. \\ & \quad - \frac{2 \sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_{l'}^{(s)} \phi_l(\mathbf{x}_i^{(t)}) \cdot \phi_{l'}(\mathbf{x}_{i'}^{(s)})}{\sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_{l'}^{(s)}} \\ & \quad \left. + \frac{\sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} \sum_{i''=1}^{n^{(s)}} H_{i'j}^{(s)} H_{i''j}^{(s)} w_{l'}^{(s)2} \phi_{l'}(\mathbf{x}_{i'}^{(s)}) \cdot \phi_{l'}(\mathbf{x}_{i''}^{(s)})}{(\sum_{l'=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_{l'}^{(s)})^2} \right\}, \end{aligned} \quad (6.10)$$

s.t., $\omega^{(t)T} \mathbf{1} = 1, U^{(t)} \mathbf{1} = \mathbf{1}$. To facilitate transfer learning of the multiple kernel metric (parameterized by $\mathbf{w}^{(s)}$) from the weak supervised task to the clustering task at hand, we define:

$$\omega_i^{(t)} w_j^{(s)} \phi_i(\mathbf{a}) \cdot \phi_j(\mathbf{b}) = \begin{cases} w_i^{(s)2} K_i(\mathbf{a}, \mathbf{b}) & , \text{ if } i = j \\ 0 & , \text{ otherwise.} \end{cases}$$

Objective (6.10) then reduces to the following ($\omega^{(t)T} \mathbf{1} = 1$ for the second term),

$$\begin{aligned} & \min_{U^{(t)}} \sum_{i=1}^{n^{(t)}} \sum_{j=1}^k \mu_{ij}^{(t)} \left\{ - \frac{2 \sum_{l=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_l^{(s)2} K_l(\mathbf{x}_i^{(t)}, \mathbf{x}_{i'}^{(s)})}{\sum_{l=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_l^{(s)}} \right. \\ & \quad \left. + \frac{\sum_{l=1}^m \sum_{i'=1}^{n^{(s)}} \sum_{i''=1}^{n^{(s)}} H_{i'j}^{(s)} H_{i''j}^{(s)} w_l^{(s)2} K_l(\mathbf{x}_{i'}^{(s)}, \mathbf{x}_{i''}^{(s)})}{(\sum_{l=1}^m \sum_{i'=1}^{n^{(s)}} H_{i'j}^{(s)} w_l^{(s)})^2} \right\} \\ & = \sum_{i=1}^n \sum_{j=1}^k \mu_{ij}^{(t)} \gamma_{ij}, \quad \text{s.t., } U^{(t)} \mathbf{1} = \mathbf{1}. \end{aligned} \quad (6.11)$$

Therefore, the cluster membership assignment involves the calculation of kernel functions $K_l(\mathbf{x}_i^{(t)}, \mathbf{x}_{i'}^{(s)})$ and $K_l(\mathbf{x}_{i'}^{(s)}, \mathbf{x}_{i''}^{(s)})$, along with the use of $H^{(s)}$ and $\mathbf{w}^{(s)}$ estimated under weak supervision, to yield the following update rule,

$$\mu_{ij}^{(t)} = \begin{cases} 1 & , \text{ if } \gamma_{ij} < \gamma_{ij'} \quad \forall j' \\ 0 & , \text{ otherwise.} \end{cases} \quad (6.12)$$

Thus the target task of MKTC to perform unsupervised single-instance clustering of $X^{(t)}$ can be done in a single step using (6.12). The cost of computing the cluster memberships from (6.12) depends on the cost of calculating all γ_{ij} . From eqn. (6.11), we observe that computing all γ_{ij} involves computing all $K_l(\mathbf{x}_i^{(t)}, \mathbf{x}_{i'}^{(s)})$ which takes $O(mkn^{(s)}n^{(t)})$ time, and computing all $K_l(\mathbf{x}_{i'}^{(s)}, \mathbf{x}_{i''}^{(s)})$ which takes $O(mkn^{(s)^2})$ times. Thus the computation cost of the target task is $O(mkn^{(s)}n^{(t)} + mkn^{(s)^2})$. This is a one-time cost only (i.e., not an iterative step), and is efficient when the size of the weak supervised dataset is smaller than the entire dataset ($n^{(s)} < n^{(t)}$), leading to an $O(mkn^{(s)}n^{(t)})$ cost. The complete method of MKTC is outlined in Algorithm 7. Combining the $O(mkn^{(s)})$ computation cost of MKMIKM and the $O(mkn^{(s)}n^{(t)})$ cost of the target task, the overall computation complexity of MKTC is $O(mkn^{(s)}n^{(t)})$.

Algorithm 7 Multiple Kernel Transfer Clustering (MKTC)

Input: Dataset $X^{(t)}$, data bags $X^{(s)}$, bag labels $Y^{(s)}$.

Output: Cluster membership matrix $U^{(t)}$.

- 1: Run Algorithm 1 with inputs $X^{(s)}$ and $Y^{(s)}$; Obtain cluster assignments $H^{(s)}$ and kernel weights $\mathbf{w}^{(s)}$.
 - 2: Compute all cluster memberships $\mu_{ij}^{(t)}$ using eq. (6.12).
-

6.5 Experiments and Results

In this section, we investigate the clustering performance of the proposed MKTC in comparison with other transfer clustering methods and the state-of-the-art multiple kernel clustering methods. We construct a transfer clustering method following our defined framework for the recently proposed weakly supervised Multi-Instance k -Means (MIKM) (Law et al., 2017) where the cluster centers are transferred from the multi-instance clustering source task of objective (6.1) to cluster the entire dataset in the target task in a single step. Among the state-of-the-art unsupervised multiple kernel clustering methods, we perform empirical comparisons with Optimal Neighborhood Kernel Clustering with Multiple Kernels (ONKCMK) (Liu et al., 2017b), Co-regularized Spectral Clustering with Improved Kernels (CRSC-IK) (Wang et al., 2017), Incomplete MKKM with Mutual Kernel Completion (MKKM-IK-MKC) (Liu et al., 2020b), Optimal Neighborhood Multiple Kernel Clustering with Adaptive Local Kernels (ONMKC-ALK) (Liu et al., 2020a), and Multiple Kernel k -Means clustering by Selecting Representative Kernels (MKKM-SRK) (Yao et al., 2020). In addition, the performance of unsupervised k -Means clustering is also compared.

6.5.1 Datasets

The datasets on which the empirical studies are conducted are listed in Table 6.1. For each dataset, multi-instance subsets for the source task clustering under weak supervision are prepared in the following manner. Datasets with three-channel images are averaged to grayscale. For the datasets coil-20, coil-100, yaleb, stl10, cifar10, and cifar100, the images

6. Transfer Clustering using Multiple Kernel Metrics Learned under Weak Supervision

resized to the size (32,32) with nearest-neighbor interpolation. All images are flattened to vectors, which are scaled by dividing each feature by its maximum value.

Table 6.1: Dataset Specifications. For each dataset the number of images N , the size of each image, and the number of clusters k is stated.

Dataset	N	Image Size	k
digits	1797	(8,8)	10
olivetti	400	(64,64)	40
umist	575	(112,92)	20
usps	11000	(16,16)	10
coil-20	1440	(128,128)	20
coil-100	7200	(128,128,3)	100
yaleb	2414	(192,168)	38
stl10	13000	(96,96,3)	10
mnist	70000	(28,28)	10
fashion	70000	(28,28)	10
cifar10	60000	(32,32,3)	10
cifar100	50000	(32,32,3)	20

Table 6.2: Specifications of the Multi-Instance Subsets Formed for each Dataset. For each Dataset the Number of Bags is b , the Number of Data Instances is n , and the Number of Dimensions is d .

Dataset	b	Range of n	d
digits	898	3110-3187	64
olivetti	200	686-744	4096
umist	287	991-1063	10304
usps	5000	17357-17588	256
coil-20	720	2474-2585	1024
coil-100	3600	12507-12756	1024
yaleb	1207	4145-4255	1024
stl10	5000	17405-17650	1024
mnist	5000	17377-17708	784
fashion	5000	17345-17581	784
cifar10	5000	17356-17700	1024
cifar100	5000	17420-17648	1024

For each dataset, the number of bags is set to $\min\{5000, n/2\}$ to ensure reduced cost of labeling for larger datasets, while minimizing oversampling for smaller datasets. The size of each bag is randomly set between a minimum of 2 and a maximum of 5, following which data instances are sampled from random classes into the multi-instance dataset $X^{(s)}$. This ensures that no bag has more than one data instance from the same class. Which classes have instances in the bag, are noted in the bag label matrix $Y^{(s)}$. In this manner, 10 sets of weakly supervised multi-instance data are prepared for each dataset, to provide robust estimations of the performance of each method in consideration. The weakly supervised multi-instance data generated for our experiments are provided at <https://github.com/Avisek20/MKTC>

along with URLs to the original datasets. The specifications of the generated multi-instance data are provided in Table 6.2.

6.5.2 Experiment Protocol

For MKTC and MIKM, the learned multiple kernel metric and cluster centers respectively on each of the 10 datasets is transferred to cluster the entire dataset. The average Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) achieved over these 10 runs is reported. On every multi-instance dataset, each method is run 10 times for a maximum of 100 iterations. For MKTC, the parameter σ_l for each Gaussian kernel is set to the $D \times \delta_l$, where $D = \max_{i,j} \|\mathbf{x}_i - \mathbf{v}_j\|^2$ and $\delta_l \in \{0.01, 0.05, 0.1, 1, 10, 50, 100\}$ (Du et al., 2015). Each kernel matrix is scaled by its maximum value, i.e., $\forall l K_l(\mathbf{x}_i, \mathbf{v}_j) := K_l(\mathbf{x}_i, \mathbf{v}_j) / \max_{i,j} K_l(\mathbf{x}_i, \mathbf{v}_j)$. k -Means is run on each dataset 10 times, with a maximum of 100 iterations per run, with an error tolerance of 10^{-5} , and the average ARI is reported. For the multiple kernel clustering methods, the standard 12 kernels are constructed following Du et al. (2015). The following parameter settings are used for the multiple kernel clustering methods based on recommendations from their original papers so that they can be evaluated in an unsupervised manner without supervised grid search over possible parameter values. For ONKCMK ρ is set to 2^{-4} and λ is set to 2^{-7} . For CRSC-IK λ is set to 0.5. For MKKM-IK-MKC λ is set to 1. For ONMKC-ALK ρ is set to 2^{-1} and γ is set to 0. For MKKM-SRK λ is set to 0.01. All multiple kernel clustering methods are run 10 times with random restarts for a maximum of 100 iterations and an error tolerance of 10^{-5} , and the average ARI is reported. The source codes of all experiments including the methods in contention are provided at <https://github.com/Avisek20/MKTC> to reproduce the results.

6.5.3 Comparison of Clustering Performances

The results of clustering the datasets are shown in Table 6.3. We observe that following the defined transfer clustering framework, the proposed MKTC achieves significantly higher average ARI in most datasets in comparison to the state-of-the-art multiple kernel clustering methods, k -Means as well as the transfer clustering method constructed from the recently proposed MIKM. MKTC obtained significantly higher average ARI in comparison to the unsupervised clustering methods: On umist, MKTC obtained 0.9327 average ARI in comparison to the highest by MKKM-IK-MKC of 0.3695; on coil-100, MKTC obtained 0.8719 average ARI in comparison to the highest average ARI of 0.4212 by k -Means; on usps, MKTC obtained 0.7327 in comparison to the highest of 0.3066 by MKKM-SRK. Only on the dataset yaleb have we observed three of the multiple kernel clustering methods to have performed well; the smaller size of the yaleb dataset may be a constraint for how effective the transfer clustering methods can be. Across all datasets, MKTC always outperforms k -Means significantly, even on the larger datasets for which the multiple kernel clustering methods could not be run due to the computation of the extremely large kernel matrices being infeasible. Thus we can conclude that MKTC, in general, is extremely competent in accurately clustering data when some supervision can be provided on a smaller multi-instance dataset. MKTC achieves its high performance at a $O(mkn^{(s)}n^{(t)})$ computational cost ($n^{(s)} < n^{(t)}$), which is significantly lower than the $O(mkn^{(t)^2})$ cost of the unsupervised multiple kernel clustering methods.

6. Transfer Clustering using Multiple Kernel Metrics Learned under Weak Supervision

Table 6.3: The Average ARI achieved by the Clustering Methods in contention. The First Six Methods are Unsupervised Clustering Methods, and the last two are Transfer Clustering Methods. Entries marked with - were Infeasible to run on the Large-sized Datasets due to the High Computation Cost of the Corresponding Method.

Methods	digits	olivetti	umist	usps	coil-20	coil-100
<i>k</i> -Means	0.6653	0.4314	0.3358	0.2858	0.6146	0.4212
ONKCMK (AAAI 2017)	0.6081	0.5353	0.3643	0.3056	0.5545	0.3925
CRSC-IK (IJCAI 2017)	0.6095	0.5235	0.3484	0.3059	0.5935	0.3693
MKKM-IK-MKC (IEEE TPAMI 2020)	0.6624	0.5663	0.3695	0.3037	0.5401	0.4128
ONMKC-ALK (IEEE TKDE 2020)	0.6081	0.0616	0.1551	0.3059	0.5695	0.2033
MKKM-SRK (IEEE TNNLS 2020)	0.6601	0.2084	0.1607	0.3066	0.1230	0.1091
MIKM (CVPR 2017)	0.8011	0.8136	0.7660	0.6331	0.7959	0.6710
MKTC	0.9072	0.8967	0.9327	0.7327	0.9007	0.8719

Methods	yaleb	stl10	mnist	fashion	cifar10	cifar100
<i>k</i> -Means	0.0137	0.0532	0.3653	0.3562	0.0334	0.0193
ONKCMK (AAAI 2017)	0.1087	0.0552	-	-	-	-
CRSC-IK (IJCAI 2017)	0.0976	0.0555	-	-	-	-
MKKM-IK-MKC (IEEE TPAMI 2020)	0.0981	0.0542	-	-	-	-
ONMKC-ALK (IEEE TKDE 2020)	0.0307	0.0550	-	-	-	-
MKKM-SRK (IEEE TNNLS 2020)	0.0113	0.0348	-	-	-	-
MIKM (CVPR 2017)	0.0120	0.0647	0.6162	0.4623	0.0384	0.0247
MKTC	0.0456	0.0720	0.6786	0.4964	0.0443	0.0276

Table 6.4: The Average ARI obtained by MIKM and MKMIKM for Weakly Supervised Multi-Instance Clustering.

Methods	digits	olivetti	umist	usps	coil-20	coil-100
MIKM (CVPR 2017)	0.9807	0.9896	0.9777	0.9202	0.9531	0.9653
MKMIKM	0.9297	0.9802	0.9760	0.9354	0.9652	0.9795

Methods	yaleb	stl10	mnist	fashion	cifar10	cifar100
MIKM (CVPR 2017)	0.1482	0.2292	0.9311	0.8059	0.1639	0.1663
MKMIKM	0.1696	0.2181	0.9208	0.7837	0.1640	0.2251

We also observe that MIKM performs better in comparison to the unsupervised clustering methods on most datasets while performing worse only in comparison to MKTC. This shows the efficacy of methods derived from the proposed transfer clustering framework, allowing them to feasibly cluster large datasets while leading to better clustering performances in comparison to the state-of-the-art unsupervised multiple kernel clustering methods.

One part of MKTC involves the MKMIKM method to perform clustering of the weakly supervised multi-instance subsets of the datasets. We can compare the clustering performances of MKMIKM with MIKM on these multi-instance subsets alone, as shown in Table 6.4. We observe that MIKM using the Euclidean distance is observed to achieve comparable ARI on the weakly supervised multi-instance datasets in comparison to MKMIKM which learns a multiple kernel metric. However on the original single-instance datasets in Table 6.3 MIKM is observed to achieve consistently lower ARI in comparison to MKTC. This indicates

that MIKM overfits the smaller multi-instance datasets, due to which its performance does not generalize to the larger single-instance datasets. In comparison MKTC is observed to obtain higher average ARIs on the single-instance datasets, while on the source tasks MKMIKM achieves comparable performances. This indicates better generalization capabilities of MKTC when learning a multiple kernel metric on a smaller subset and transferring it to the target task where a larger dataset is to be clustered.

We can also observe how learning the multiple kernel metric under multi-instance weak supervision by MKMIKM has affected the clustering performance of MKTC, by directly comparing the performance of MKTC with an Unsupervised Multiple-Kernel k -Means (us-MKKM), which also learns a multiple kernel metric while clustering, but does not have access to the weakly supervised multi-instance subset that MKTC has. For us-MKKM we optimize a modified objective (6.4) where $H^{(s)}$ is not constrained to $Q^{U(s)}$, but instead the objective function is optimized over $H \in \{0, 1\}^{n \times k}, H\mathbf{1} = \mathbf{1}$. This modified objective is optimized over the entire target dataset, under the same experimental protocol and the average ARI is compared with that of MKTC. The results are shown in Table 6.5, where we observe that MKTC obtains significantly higher average ARIs compared to us-MKKM, demonstrating how for MKTC learning the multiple kernel metric under weak supervision has led to significant improvements in clustering across all datasets.

Table 6.5: The Average ARI obtained by Unsupervised Multiple Kernel k -Means (us-MKKM) in comparison with MKTC.

Methods	digits	olivetti	umist	usps	coil-20	coil-100
us-MKKM	0.5148	0.3645	0.3284	0.2848	0.5703	0.4427
MKTC	0.9072	0.8967	0.9327	0.7327	0.9007	0.8719
Methods	yaleb	stl10	mnist	fashion	cifar10	cifar100
us-MKKM	0.0169	0.0569	0.3728	0.3489	0.0333	0.0193
MKTC	0.0456	0.0720	0.6786	0.4964	0.0443	0.0276

6.5.4 Execution Times of Multiple Kernel Clustering Methods

In the previous section we observed the significantly better performance of MKTC achieved at a lower $O(mkn^{(s)}n^{(t)})$ computational cost, compared to the state-of-the-art unsupervised multiple kernel clustering methods which have higher $O(mkn^{(t)^2})$ costs. The effect of the computational complexity on the actual execution times can be observed in Table 6.6. We measure the average execution time for one run of each method, over 10 runs on every dataset. For all the methods, we compute all kernel and distances parallelly over 48 processor threads of an otherwise idle AMD Ryzen Threadripper 3960X with 128GB RAM. On the smaller datasets of olivetti and umist, the multiple kernel clustering methods can have lower execution times when they converge in 1 iteration while MKTC iterates over more iterations. However in general we can observe MKTC outperform the multiple kernel clustering methods, especially when the size of the datasets increase, as is observed for usps, coil-100, and stl10. For the largest datasets of mnist, fashion, cifar10, and cifar100, the unsupervised multiple kernel clustering methods cannot be run due to the high memory size required to compute the kernel matrices.

Table 6.6: The Average Execution Time (in seconds) observed for the Multiple Kernel Clustering Methods. All Kernel and Distance Computations are done Parallely over 48 Processor Threads that are otherwise Idle. Entries marked with - were Infeasible to run on the Large-sized Datasets due to the High Computation Cost of the Corresponding Method.

Methods	digits	olivetti	umist	usps	coil-20	coil-100
ONKCMK (AAAI 2017)	8.31E+00	3.62E-01	8.03E-01	1.08E+03	5.22E+00	3.79E+02
CRSC-IK (IJCAI 2017)	2.54E+01	1.94E+00	3.46E+00	3.75E+03	1.49E+01	1.18E+03
MKKM-IK-MKC (IEEE TPAMI 2020)	1.98E+02	6.84E+00	1.36E+01	3.38E+04	9.80E+01	1.01E+04
ONMKC-ALK (IEEE TKDE 2020)	1.14E+01	8.11E+00	1.74E+00	1.58E+03	6.36E+00	4.80E+02
MKKM-SRK (IEEE TNNLS 2020)	1.03E+01	6.41E-01	1.62E+00	9.89E+02	5.78E+00	4.94E+02
MKTC	1.41E+00	2.59E+00	5.12E+00	3.64E+01	1.40E+00	6.96E+01

Methods	yaleb	stl10	mnist	fashion	cifar10	cifar100
ONKCMK (AAAI 2017)	1.70E+01	1.78E+03	-	-	-	-
CRSC-IK (IJCAI 2017)	4.83E+01	6.17E+03	-	-	-	-
MKKM-IK-MKC (IEEE TPAMI 2020)	3.10E+02	7.31E+04	-	-	-	-
ONMKC-ALK (IEEE TKDE 2020)	2.58E+01	2.42E+03	-	-	-	-
MKKM-SRK (IEEE TNNLS 2020)	1.88E+01	2.00E+03	-	-	-	-
MKTC	3.30E+00	6.99E+01	7.01E+01	7.74E+01	1.02E+02	1.55E+02

6.5.5 Effect of Number of Bags and Bag Size

For the source task of MKMIKM, part of the multi-instance setup involves deciding on the maximum number of bags $n^{(s)}$, and the maximum bag size permitted $\max_i n_i$. As discussed in Section 6.5.1, in the experiments we allowed the maximum number of bags $n^{(s)}$ to be 5000, and the maximum permitted bag size $\max_i n_i$ to be 5. We can investigate the effect these two constraints have on the overall clustering performance. We conduct experiments on the six largest datasets we considered: usps, mnist, fashion, stl10, cifar10, and cifar100, to minimize multiple sampling of the same data instance. First we investigate the effect of the maximum bag size, for which we vary $n^{(s)}$ from 3000 to 8000 in increments of 1000, and generate 10 subsets of each dataset for each of the maximum bag sizes. We note the average ARI achieved by MKTC over 10 runs across the 10 subsets, shown in Table 6.7. We observe that across increasing values of $n^{(s)}$, there is very little variability in the average ARI achieved.

Table 6.7: The Average ARI observed for MKTC as the Total Number of Bags ($n^{(s)}$) is Increased.

$n^{(s)}$	usps	stl10	mnist	fashion	cifar10	cifar100
3000	0.7271	0.0720	0.6797	0.4964	0.0439	0.0274
4000	0.7301	0.0717	0.6817	0.4963	0.0435	0.0271
5000	0.7327	0.0720	0.6786	0.4964	0.0443	0.0276
6000	0.7322	0.0718	0.6812	0.4960	0.0439	0.0274
7000	0.7333	0.0713	0.6820	0.4966	0.0440	0.0271
8000	0.7324	0.0718	0.6801	0.4964	0.0442	0.0272

Similarly, we also study the effect that the maximum permitted size of a bag $\max_i n_i$ has on the overall clustering performance. We generate 5000 bags, where each bag is generated with a minimum of size 2 and the maximum that ranges from 4 to 9. Similar to the previous

study, we conduct the study on the same six datasets and generate 10 subsets of each dataset for each of the maximum permitted sizes of the bag. The average ARI achieved by MKTC over 10 runs across the 10 subsets is shown in Table 6.8. We observe that as the maximum permitted size is increased, the average ARI achieved by MKTC generally decreases. This is because, for larger bag sizes, MKMIKM has the more difficult task of assigning a single data instance among a larger set of data instances to a possible candidate cluster. This is easier to do when the maximum permitted size is smaller, therefore encouraging the setup of smaller bag sizes for MKMIKM, which will also keep the overall computation complexity to a minimum and lead to smaller execution times.

Table 6.8: The Average ARI observed for MKTC as the Maximum Possible Size of a Single Bag ($\max_i n_i$) is Increased.

max n_i	usps	stl10	mnist	fashion	cifar10	cifar100
4	0.7358	0.0747	0.6855	0.5007	0.0457	0.0279
5	0.7327	0.0720	0.6786	0.4964	0.0443	0.0276
6	0.7251	0.0702	0.6734	0.4931	0.0426	0.0265
7	0.7212	0.0686	0.6698	0.4881	0.0410	0.0261
8	0.7119	0.0659	0.6628	0.4527	0.0388	0.0255
9	0.6839	0.0634	0.6291	0.4158	0.0383	0.0248

The above two observations show that (i) the number of bags in the source task can be kept to a minimum, and (ii) the size of each bag can be kept small. Therefore MKTC can effectively cluster a dataset when small multi-instance subsets are constructed in the source task, leading to low computational costs and faster execution times.

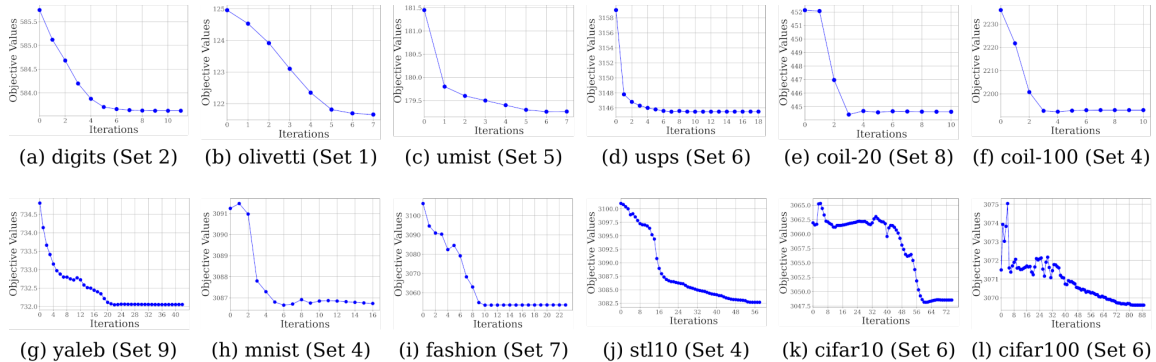


Figure 6.2: Empirical convergence of MKMIKM observed over randomly selected multi-instance subsets for each of the benchmark computer vision datasets. Each plot shows the change in MKMIKM’s objective function value with increase in number of iterations.

6.5.6 Empirical Convergence of MKMIKM

While the target task of MKTC is done in a single step, the source task involves the MKMIKM method which is an alternating optimization algorithm. We show empirical convergence of MKMIKM in Figure 6.2, where plots for the change in objective function value over iterations are shown for a randomly selected multi-instance subset of the datasets considered. We

observe that the objective function value always shows eventual numerical convergence, with occasional increases towards the start of the run only on the larger and more complex datasets. This justifies the check on changes in H between successive iterations to declare numerical convergence.

6.6 Discussions

In this chapter, we proposed the Multiple Kernel Transfer Clustering (MKTC) method that can be described in terms of two tasks, a source task where a multiple kernel metric is learned, and a target task where the multiple kernel metric is transferred to cluster a dataset. The source task creates a weakly supervised multi-instance subset of the dataset. For this, we devised a new Multiple Kernel Multi-Instance k -Means (MKMIKM) clustering objective along with an alternating optimization method to cluster the multi-instance subset while learning a multiple kernel metric under weak supervision. In the target task, MKTC transfers the multiple kernel metric learned by MKMIKM to perform unsupervised single-instance clustering of the entire dataset in a single step. The advantage of using a multi-instance setup for the source task is that it requires reduced labeling effort to guide the learning of the multiple kernel metric. Our formulations of MKMIKM lead to a low $O(mkn^{(s)})$ computation cost per iteration, and the single clustering step of the target task requires $O(mkn^{(s)}n^{(t)} + mkn^{(s)2})$ time, leading to an overall $O(mkn^{(s)}n^{(t)})$ cost for MKTC where $n^{(s)} < n^{(t)}$, which is a significant reduction in comparison to the state-of-the-art multiple kernel clustering algorithms which require $O(mkn^{(t)2})$ time. Experiments over benchmark computer vision datasets indicate that MKTC achieves significant performance improvement when compared to the state-of-the-art unsupervised multiple-kernel clustering methods and other transfer clustering methods. The commendable level of performance as well as the low computational costs make the proposed MKTC well suited for the clustering of larger datasets.

Chapter 7

Conclusion

In this thesis, we identified four factors that affect the performance of center-based clustering methods: the number of clusters to be identified, the distance metric used, the features utilized, and the nature of the clusters identified. We studied each of these factors and proposed methods for each that would lead to low-cost center-based clustering. In addition, we considered clustering in an unsupervised setting to be a limitation, and studied clustering under low effort supervision to improve clustering performances. In this chapter, we summarize the contributions of our studies, followed by a discussion on interesting future directions that our studies indicate towards.

7.1 Contributions of the Thesis

For unsupervised center-based clustering, we studied four factors that affect its performance, namely the number of clusters to be identified, the distance metric used, the features utilized, and the nature of the clusters identified. Our first study in Chapter 2 focused on efficiently estimating the number of clusters. We proposed two cluster number estimation methods based on two possible definitions of a cluster, one where clusters are of equal size, and one where clusters are well-separated. When clustering at successive number of clusters, we observed that there can be a large reduction in the minimum distance between cluster centers when proper cluster structures are identified. This formed the basis of our proposed cluster number estimation methods. We studied the cluster number estimation capabilities of several popular and recent cluster number estimation methods, and observed that our proposed methods performed among the best, while being among the methods that had the lowest computation costs.

Our next study was on the distance metric used, for which in Chapter 3 we proposed using a kernel metric in a center-based clustering method called k -Harmonic Means. We formulated an objective function for a kernelized general Fuzzy c -Means, and showed that from the objective function one could derive the objective functions for Kernel k -Harmonic Means, Kernel Fuzzy c -Means, and Kernel k -Means. We tested our proposed Kernel k -Harmonic Means algorithm against other kernel clustering methods, and observed its competent performance. In the situation where a large number of clusters were present, we observed Kernel k -Harmonic Means to obtain significantly improved clustering performance in comparison to other kernel clustering methods.

We followed this study by our study on model selection for sparse clustering methods in Chapter 4. Sparse clustering methods like Sparse k -Means and Sparse Fuzzy c -Means achieve feature selection by imposing an ℓ_1 -norm constraint on the feature weights, and require a further parameter for the upper bound to the ℓ_1 -norm, which controls the sparsity of the features. The upper bound has traditionally been controlled through a computationally intensive process requiring further clustering of randomly generated datasets of the same size as the original dataset. We proposed deriving expressions for the Bayesian Information Criteria for Sparse k -Means and Sparse Fuzzy c -Means that can be used to select the degree of sparsity. Thus Sparse k -Means and Sparse Fuzzy c -Means can be run for different candidate degrees of sparsity, and our derived expressions for the Bayesian Information Criterion can be used to select an appropriate degree of sparsity while requiring low computation costs. Extensive experiments for both Sparse k -Means and Sparse Fuzzy c -Means in comparison to several other possible choices cluster validation as well showed that using our derived expressions of Bayesian Information Criterion led to significantly improved performance in sparse clustering.

For the final factor of the nature of clusters one can identify, we studied fuzzy clustering methods to identify overlapped clusters. We proposed an evolutionary multi-objective fuzzy clustering method in Chapter 5 that identified fuzzy clusterings at different levels of fuzziness in order to identify clusters with different levels of overlap. Our proposed method achieves this by optimizing two contradictory objectives in an evolutionary multi-objective setting. The first objective minimizes the sum of intra-cluster variances, and prefers the identification of discrete clusters. The second objective maximizes the entropy of cluster memberships, and thus tries to identify completely overlapped clusters. Optimizing both these objectives in a multi-objective setting leads to a Pareto front of fuzzy clusterings at different levels of fuzziness. Our experiments show the quality of the Pareto front, which contains wide and diverse solutions of fuzzy clusterings at different levels of fuzziness. We also propose a method to identify an appropriate fuzzy clustering for comparison under a discrete cluster evaluation measure, and investigate its performance for real and synthetic datasets, and observe competent performances of our proposed method.

Finally, we considered clustering under completely unsupervised conditions to be a limitation, and proposed a method of clustering under some degree of supervision. We proposed a multiple kernel transfer clustering method in Chapter 6 which learns a multiple kernel metric under a weak supervision setup, and transfers the learned metric to cluster a dataset in an unsupervised setup. In greater detail, our method can be described in terms of two tasks. In the source task, we constructed a weakly supervised multi-instance setup where a set of data instances are assigned a vector of possible cluster labels. This setup requires less labeling effort in comparison to a fully supervised setup, where labels need to be assigned to every data instance. In this weakly supervised multi-instance setup we performed clustering while simultaneously learning a multiple kernel metric. This multiple kernel metric is transferred to a target task involving the unsupervised single-instance clustering of a dataset. Experiments on large computer vision datasets show that our proposed method leads to significant improvements in clustering performance, in comparison to other weak supervision methods as well as state-of-the-art unsupervised multiple kernel clustering methods.

In this thesis we therefore examined the field of center-based clustering from different perspectives, examining different factors that can lead to improved clustering performance while keeping the computational cost as low as possible, so that it can be applied to large

datasets which is commonplace nowadays across all application domains. Our consideration of the unsupervised setting and our investigations in clustering under some degree of supervision leads us to recommend further empirical as well as theoretical studies in the area of clustering under some degree of supervision. Our investigations lead to interesting new directions of research that are possible, which are discussed in the following section.

7.2 Future Directions of Research

In this section we consider possible directions of research that the investigations in this thesis lead to.

- **Statistical tests for cluster number estimation:** In our studies on cluster number estimation in Chapter 2, we studied several cluster validity indices as well as proposed two approaches to estimate the number of clusters. In this setup, estimating the cluster number depends on specific rules, such as considering the maximum or minimum value of an estimation method. A more suitable approach could be by using a statistical test, which provides a p -value accompanying the test results, which can be significantly more informative when deciding on the number of clusters. Such a test can be used on its own, or can be a part of a clustering method that progressively divides or joins clusters, or even in an evolutionary clustering method. The test needs to be of low computation cost so that it can be used frequently during a clustering process. [Efimov et al. \(2019\)](#) is an interesting example of a recently proposed statistical test, however it also has a high cost of computation. A statistical test with low computation cost will definitely be of great use across different clustering methodologies and will have significant applicability.
- **On more informed applications of multiple kernel metrics:** In chapter 3 we incorporated a kernel metric in a center-based clustering method to improve clustering performance. One limitation that we faced was the requirement of providing values to the kernel parameter. Subsequent works on multiple kernel clustering circumvent the necessity of providing exact values to the kernel parameter, by considering the linear combination of a set of possible kernels with fixed parameter values ([Du et al., 2015](#)). A number of interesting questions arise when considering this convention. Can the parameter values of base kernels of different types (Gaussian, Polynomial, etc) be learnt reliably in a multiple kernel setup? One can then consider only one type of such kernels and learn their parameters, instead of relying on a number of fixed variants of each kernel. If this question is difficult to answer, then one can also investigate whether there is a set of kernels that perform best in a multiple kernel setup. A related investigation can also involve considering a really large number of base kernels, and considering their sparse combination as the final multiple kernel metric.
- **Co-clustering with sparse feature selection:** We observed efficient feature selection while clustering was possible in Chapter 4. The ideas discussed can be extended to perform co-clustering of data ([Salah et al., 2016](#)), where both data instances and features are clustered simultaneously, and sparse feature selection can be performed in a co-clustering setup. An explainable co-clustering method ([Moshkovitz et al., 2020](#)) would

also be of significant applicability, since feature selection easily lends to interpretable models.

- **The effect of dimensionality on evolutionary multi-objective clustering:** A notable observation from the results of our experiments in Chapter 5 was that ECM-NSGA-II performed better for lower dimensional datasets whereas ECM-MOEA/D performed better for higher dimensional datasets. A theoretical analysis of this behaviour would be a challenging yet interesting study, which requires an analysis of how the way each method operates is affected by the dimensionality of the data. Such an analysis should be done in the general context of the wide range of evolutionary multi-objective methods that exist to analyze how the dimensionality of the data in the clustering context affects their performance.
- **Weakly supervised balanced clustering:** In Chapter 5 we observed a variant of center-based clustering based on definitions of fuzzy clusters to identify overlapped clusters. Another variant that has been receiving attention recently is balanced clusters, where the cluster sizes are controlled by the clustering algorithm (Jitta and Klami, 2018; Lin et al., 2019).
- **Deep multi-view clustering under some degree of supervision:** Recently there has been great success in deep clustering, which involves non-linear projections of data to spaces where it can be clustered effectively (Chang et al., 2020; Lv et al., 2021). This can also be viewed as a non-linear feature extraction procedure that is done simultaneously while clustering. The non-linear projection can be guided using information from multiple views (Chen et al., 2020a; Wei et al., 2020; Wang et al., 2021). In such a setup, adding some degree of supervision as discussed in Chapter 6 may help to improve the non-linear feature extraction procedure as well as perform clustering.
- **Convex Clustering under weak supervision:** An interesting area of clustering is convex clustering, on which theoretical guarantees on the performance can be provided (Sun et al., 2021). Convex clustering under weak supervision may lead to methods with theoretical guarantees as well as good empirical performances, and effective approximations for application to large datasets can be investigated as well (Yuan et al., 2018).

Appendix A

Supplementary for Chapter 2

A.1 Cluster Number Identification Methods

In this section, we provide the full mathematical formulation for all cluster number identification methods that we have used in the experiments in Section 2.5. Throughout this section, we use the following notations: $\mathbf{x}_1, \dots, \mathbf{x}_n$ are n data points, where $\mathbf{x}_i \in \mathbb{R}^d$. Clusters are represented as C_1, \dots, C_k , and the centers of each of k clusters are $\mathbf{v}_1, \dots, \mathbf{v}_k$. Fuzzy memberships of \mathbf{x}_i to cluster C_j are represented by i_j , with m as the parameter for degree of fuzziness. n_j is the sum of fuzzy memberships to cluster j , i.e., $n_j = \sum_{i=1}^n i_j$.

1. Akaike Information Criterion (AIC) (Akaike, 1974):

$$AIC(k) = \sum_{j=1}^k \left\{ (-2(n_j \log(n_j) - (n_j - n) - \frac{n_j d}{2} \log(2\pi \Delta_j) - \frac{d(n_j - 1)}{2})) \right\} + 2k,$$

$$\text{where, } \Delta_j = \sum_{\mathbf{x}_i \in C_j} \frac{\|\mathbf{x}_i - \mathbf{v}_j\|^2}{d/(n-k)}.$$

\hat{k} : The Elbow Method.

Minimum number of clusters identifiable: 2.

2. Bayesian Information Criterion (BIC) (Schwarz, 1978):

$$BIC(k) = \sum_{j=1}^k \left\{ ((n_j \log(n_j) - (n_j - n) - \frac{n_j d}{2} \log(2\pi \Delta_j) - \frac{d(n_j - 1)}{2})) \right\} - \frac{k \log(n)(d+1)}{2},$$

$$\text{where, } \Delta_j = \sum_{\mathbf{x}_i \in C_j} \frac{\|\mathbf{x}_i - \mathbf{v}_j\|^2}{d/(n-k)}.$$

\hat{k} : Maximum of BIC .

Minimum number of clusters identifiable: 2.

3. Caliński and Harabasz (CH) Index (Caliński and Harabasz, 1974):

$$CH(k) = \frac{n-k}{k-1} \frac{\sum_{j=1}^k |C_j| \cdot \|\mathbf{v}_j - \bar{\mathbf{x}}\|^2}{\sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2},$$

\hat{k} : Maximum of CH .

Minimum number of clusters identifiable: 2.

4. Classification Entropy (CE) (Bezdek, 1975):

$$CE(k) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c \mu_{ij} \log \mu_{ij}.$$

\hat{k} : Minimum of CE .

Minimum number of clusters identifiable: 2.

5. Compose-Within-Between (Rezaee et al., 1998):

$$CWB(k) = \alpha Scat(k) + Dis(k).$$

$$\text{Here, } Scat(k) = \frac{1}{k \|\sigma_X\|} \sum_{j=1}^k \|\sigma_j\|,$$

$$\text{where, } \sigma_j = \frac{1}{n} \sum_{i=1}^n \mu_{ij} (\mathbf{x}_i - \mathbf{v}_j)^2,$$

$$\text{and, } \sigma_X = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})^2.$$

$$\text{And, } Dis(c) = \frac{\max_{j \neq t} \|\mathbf{v}_j - \mathbf{v}_t\|}{\min_{j \neq t} \|\mathbf{v}_j - \mathbf{v}_t\|} \sum_{j=1}^k \left(\sum_{l=1}^k \|\mathbf{v}_j - \mathbf{v}_l\| \right)^{-1}.$$

Also, $\alpha = Dis(k_{max})$.

\hat{k} : Minimum of CWB.

Minimum number of clusters identifiable: 2.

6. Davies-Bouldin (DB) Index (Davies and Bouldin, 1979):

$$DB(k) = \frac{1}{k} \sum_{j=1}^k \max_{j \neq l} \{DB_{jl}\},$$

$$\text{where, } DB_{jl} = \frac{\sqrt{\frac{\sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2}{|C_j|}} + \sqrt{\frac{\sum_{\mathbf{x}_i \in C_l} \|\mathbf{x}_i - \mathbf{v}_l\|^2}{|C_l|}}}{\|\mathbf{v}_j - \mathbf{v}_l\|}.$$

\hat{k} : The Elbow Method.

Minimum number of clusters identifiable: 1.

7. Dunn Index (Dunn, 1973):

$$Dunn(k) = \min_{\mathbf{x}_i \in C_l} \min_{\mathbf{x}_{i'} \in C_j, j \neq l} \left(\frac{\|\mathbf{x}_i - \mathbf{x}_{i'}\|}{\max_j \max_{\mathbf{x}_i, \mathbf{x}_{i'} \in C_j} \|\mathbf{x}_i - \mathbf{x}_{i'}\|} \right).$$

\hat{k} : Maximum of *Dunn*.

Minimum number of clusters identifiable: 2.

8. Knee Point Method: Discussed in Section 2.2 of Chapter 2.

9. Fukuyama Sugeno (FS) Index (Fukuyama and Sugeno, 1989)

$$FS(k) = \sum_{i=1}^n \sum_{j=1}^k \mu_{ij}^m \|\mathbf{x}_j - \mathbf{v}_i\|^2 - \sum_{j=1}^k \|\mathbf{v}_i - \bar{\mathbf{v}}\|^2,$$

$$\text{where, } \bar{\mathbf{v}} = \frac{1}{k} \sum_{j=1}^k \mathbf{v}_j.$$

\hat{k} : The Elbow Method.

Minimum number of clusters identifiable: 1.

10. Fuzzy Hypervolume (FHV) (Dave, 1996):

$$FHV(V) = \sum_{j=1}^k \left(\frac{\sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2}{\sum_{i=1}^n \mu_{ij}^m} \right)^{1/2}.$$

\hat{k} : Minimum of *FH*.

Minimum number of clusters identifiable: 1.

11. Gap Statistic (Tibshirani et al., 2001):

The estimation of cluster number follows a 1-standard-error rule:

$$\hat{k} = \text{the smallest } k \text{ such that } Gap(k) \geq Gap(k+1) - s_{k+1},$$

$$\text{where } Gap(k) = \frac{1}{B} \sum_{b=1}^B \{\log(W_{kb}^*) - \log(W_k)\},$$

$$\text{and, } s_k = \sqrt{1 + \frac{1}{B} \left\{ \frac{1}{B} \sum_{b=1}^B \{\log(W_{kb}^*) - \bar{l}\}^2 \right\}^{1/2}}.$$

Here W_{kb}^* is the average of the sum of intra-cluster pairwise distances for the b -th reference distribution, and W_k is the sum of the intra-cluster pairwise distances for the

data set, and,

$$\bar{l} = \frac{1}{B} \sum_{b=1}^B \log(W_{kb}^*).$$

Minimum number of clusters identifiable: 1.

12. Halkidi Vazirgannis (HV) Index ([Halkidi and Vazirgiannis, 2001](#)):

$$S_Dbw(k) = \text{Scatter}(k) + \text{Dens_Bw}(k),$$

$$\text{where, } \text{Scatter}(k) = \frac{1}{k} \frac{\sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \frac{\|\mathbf{x}_i - \mathbf{v}_j\|^2}{C_j}}{\sum_{i=1}^n \frac{\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}{n}},$$

$$\text{and } \text{Dens_Bw}(k) = \frac{1}{k(k-1)} \sum_{j=1}^k \left(\sum_{l=1, l \neq j}^k \frac{\text{Dens}(\frac{\mathbf{v}_j + \mathbf{v}_l}{2})}{\max\{\text{Dens}(\mathbf{v}_j), \text{Dens}(\mathbf{v}_l)\}} \right),$$

$$\text{and, } \text{Dens}(\mathbf{v}_j) = \sum_{i=1}^n f(\mathbf{x}_i, \mathbf{v}_j),$$

$$\text{and } f(\mathbf{x}_i, \mathbf{v}_j) = \begin{cases} 0 & , \text{ if } \|\mathbf{x}_i - \mathbf{v}_j\| > \text{stdev} , \\ 1 & , \text{ otherwise.} \end{cases},$$

where stdev is the average standard deviation of the clusters.

hatk: Minimum of S_Dbw.

Minimum number of clusters identifiable: 2.

13. Hartigan ([Hartigan, 1985](#)):

$$\text{Hart}(k) = (n - k - 1) \left(\frac{W_k}{w_{k+1}} - 1 \right),$$

$$\text{where, } W_k = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2.$$

\hat{k} : The Elbow Method.

Minimum number of clusters identifiable: 1.

14. I Index ([Maulik and Bandyopadhyay, 2002](#)):

$$I(V) = \left(\frac{1}{k} \times \frac{E_1}{E_k} \times D_k \right)^p,$$

$$\text{where, } E_k = \sum_{j=1}^k \sum_{i=1}^n \mu_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|,$$

$$\text{and, } D_k = \max_{i,j=1}^k \|\mathbf{v}_i - \mathbf{v}_j\|.$$

\hat{k} : Maximum of I .

Minimum number of clusters identifiable: 2.

15. Jump Method (Sugar and James, 2003):

$$Jump(k) = W_k^{-Y} - W_{k-1}^{-Y},$$

where, $W_k = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2$.

Here Y is set to 1 (Sugar and James, 2003).

\hat{k} : Maximum of $Jump$.

Minimum number of clusters identifiable: 1.

16. Modified Partition Coefficient (MPC) (Dave, 1996):

$$MPC(k) = 1 - \frac{k}{k-1} \left(1 - \sum_{i=1}^n \sum_{j=1}^k \frac{\mu_{ij}^2}{n}\right).$$

\hat{k} : Maximum of MPC .

Minimum number of clusters identifiable: 2.

17. Partition Coefficient (PC) (Dave, 1996):

$$PC(k) = \sum_{i=1}^n \sum_{j=1}^k \frac{\mu_{ij}^2}{n}.$$

\hat{k} : Maximum of PC .

Minimum number of clusters identifiable: 2.

18. Partition Index (PI) (Bensaid et al., 1996):

$$PI(k) = \sum_{j=1}^k \frac{\sum_{i=1}^n \mu_{ij}^n \|\mathbf{x}_i - \mathbf{v}_j\|^2}{\sum_{i=1}^n \mu_{ij} \sum_{l=1}^k \|\mathbf{v}_j - \mathbf{v}_l\|^2}.$$

\hat{k} : The Elbow Method.

Minimum number of clusters identifiable: 2.

19. PBMF (Pakhira et al., 2004):

$$PBMF(k) = \left(\frac{1}{k} \times \frac{E_1}{J_m} \times D_k \right)^2,$$

$$\text{where, } J_m(U, Z) = \sum_{i=1}^n \sum_{j=1}^k \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|,$$

$$\text{and, } E_1 = \sum_{i=1}^n \|\mathbf{x}_i - \bar{\mathbf{x}}\|,$$

$$\text{and, } D_k = \max_{i,j=1}^k \|\mathbf{v}_i - \mathbf{v}_j\|.$$

\hat{k} : Maximum of $PBMF$.

Minimum number of clusters identifiable: 2.

20. PCAES (Wu and Yang, 2005):

$$PCAES(k) = \sum_{j=1}^k \sum_{i=1}^n \mu_{ij}^2 / \mu_M - \sum_{j=1}^k \exp(-\min_{l \neq j} \{\|\mathbf{v}_j - \mathbf{v}_l\|^2\} / \beta_T),$$

$$\text{where, } \mu_M = \min_{1 \leq j \leq k} \left(\sum_{i=1}^n \mu_{ij}^2 \right),$$

$$\text{and, } \beta_T = \frac{\sum_{j=1}^k \|\mathbf{v}_j - \bar{\mathbf{v}}\|^2}{k}.$$

\hat{k} : Maximum of $PCAES$.

Minimum number of clusters identifiable: 2.

21. Prediction Strength (Tibshirani and Walther, 2005):

Prediction Strength divides the data set into c folds, and in a manner similar to cross-validation, it forms a training set from $c - 1$ folds and a test set from the remaining fold. It then clusters the training set X_{train} and test set X_{test} . The cluster centers from the clustering of X_{train} are used to assign labels to X_{test} . A co-occurrence matrix $D[C(X_{train}, k), X_{test}]_{n \times n}$ compares the clustering labels of the test set from the test centers with those from the training centers, and notes which points are assigned to the same cluster. Then,

$$PS(k) = \text{cv-avg} \min_j \frac{1}{n_j(n_j-1)} \sum_{i \neq i' \in C_j} 1(D[C(X_{train}, k), X_{test}]_{ii'} = 1).$$

22. RLWY Index (Ren et al., 2016):

$$RLWY(k) = \sum_{j=1}^k \frac{\frac{1}{n_j} \sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2 + \frac{1}{k} \|\mathbf{v}_k - \bar{\mathbf{v}}\|^2}{\frac{1}{k-1} \sum_{l=1}^k \|\mathbf{v}_l - \mathbf{v}_j\|^2}$$

\hat{k} : Minimum of *RLWY*.

Minimum number of clusters identifiable: 2.

23. Silhouette Index ([Rousseeuw, 1987](#)):

$$sil(k) = \frac{1}{n} \sum_{i=1}^n s_i,$$

where for all $\mathbf{x}_i \in C_j$,

$$a_i = d(\mathbf{x}_i, C_j), \text{ and } b_i = \min_{l \neq j} d(\mathbf{x}_i, C_l),$$

$$\text{and, } s_i = \begin{cases} \frac{b_i - a_i}{\max\{b_i, a_i\}} & , \text{ if } |C_j| > 1, \\ 1 & , \text{ if } |C_j| = 1 \end{cases}.$$

\hat{k} : Maximum of *sil*.

Minimum number of clusters identifiable: 2.

24. Slope Statistic ([Fujita et al., 2014](#)):

$$slope(k) = -[s(k+1) - s(k)]s(k)^p,$$

where, $s(k) = \frac{1}{n} \sum_{i=1}^n s_i$,

and for all $\mathbf{x}_i \in C_j$,

$$a_i = d(\mathbf{x}_i, C_j), \text{ and } b_i = \min_{l \neq j} d(\mathbf{x}_i, C_l),$$

$$\text{and, } s_i = \begin{cases} \frac{b_i - a_i}{\max\{b_i, a_i\}} & , \text{ if } |C_j| > 1, \\ 1 & , \text{ if } |C_j| = 1 \end{cases}.$$

\hat{k} : Maximum of *slope*.

Minimum number of clusters identifiable: 2.

25. Xie-Beni (XB) Index ([Xie and Beni, 1991](#)):

$$XB(k) = \frac{\sum_{j=1}^c \sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2}{n \min_{j,l} \|\mathbf{v}_j - \mathbf{v}_l\|^2}.$$

\hat{k} : Minimum of *XB*.

Minimum number of clusters identifiable: 2.

26. Xu Index (Xu, 1997):

$$Xu(k) = d \log \left(\sqrt{\frac{1}{n^2 d} \sum_{j=1}^c \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2} \right) + \log(k).$$

\hat{k} : Minimum of Xu .

Minimum number of clusters identifiable: 2.

27. ZXF Index (Zhao et al., 2009b):

$$ZXF(k) = \frac{k * \sum_{j=1}^c \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|}{\sum_{j=1}^c |C_j| \times \|\mathbf{v}_j - \bar{\mathbf{x}}\|}.$$

\hat{k} : Minimum of ZXF .

Minimum number of clusters identifiable: 2.

28. SC Index (Rezaee, 2010):

$$SC(k) = \frac{Sep(k)}{\max_k Sep(k)} + \frac{Comp(k)}{\max_k Comp(k)},$$

$$\text{and, } Sep(k) = \frac{2}{k(k-1)} \sum_{j \neq l}^k S_{rel}(C_j, C_l),$$

$$\text{and, } S(C_j, C_l) = \sum_{i=1}^n \min(\mu_{ij}, \mu_{il}) \times h(\mathbf{x}_i),$$

$$\text{and, } h(\mathbf{x}_i) = - \sum_{j=1}^k \mu_{ij} \log(\mu_{ij}).$$

\hat{k} : Minimum of SC .

Minimum number of clusters identifiable: 2.

A.2 Validation of cluster number estimation methods

We perform validation of each cluster number estimation method on 12 data sets, shown in Figure 2.6. The results of each cluster number estimation method are listed in Table A.1. The cluster number estimation methods are indexed according to the Table 2.1. The last two columns are the proposed methods Last Leap (LL) and Last Major Leap (LML).

Table A.1: The Estimated Number of Clusters for each of the Cluster Number Estimation Methods, on the 12 Validation Data Sets.

Sl. No.	k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	LL	LML
1	1	8	4	4	2	4	7	17	4	4	1	2	15	4	2	4	4	2	4	8	9	5	4	4	4	9	17	4	20	1	1
2	2	6	2	2	2	2	5	2	2	2	2	2	2	2	2	2	2	2	5	2	2	2	2	2	2	2	2	5	10	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	3	3	3	3	3	3	3	2	3	3	3	3	3	4	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3
5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
6	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7	4	4	4	4	4	4	4	4	3	3	4	4	4	3	4	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4
8	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
10	6	6	6	6	6	6	6	6	6	5	6	6	6	4	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
11	6	6	6	6	6	6	6	6	6	4	6	6	6	6	6	6	6	6	6	6	6	6	4	6	6	6	6	6	6	6	6
12	3	6	8	3	2	2	5	12	3	3	2	6	13	3	3	3	2	2	6	3	2	3	2	2	2	2	13	6	13	2	3

A.3 Results on Real Datasets

Table A.2 shows the estimated number of clusters for each cluster number estimation method. Each column is indexed by a number according to the number of the index as described in the previous section. The last two columns are the results of the proposed methods LL and LML.

Table A.2: The Estimated Number of Clusters on Real-world Data Sets.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	LL	LML
banknote	13	37	2	2	8	11	35	6	5	1	1	38	6	3	35	2	2	9	9	4	7	3	38	2	2	36	38	9	2	2
echo	5	11	2	2	8	5	2	3	3	1	3	10	4	3	2	2	2	3	3	5	2	2	11	2	2	8	11	3	1	1
iris	6	8	3	2	2	5	13	3	3	2	4	13	3	3	3	2	2	6	3	2	3	2	12	2	2	2	13	6	2	3
seeds	6	15	3	2	3	5	12	3	3	1	3	12	3	3	3	2	2	6	3	3	4	2	15	2	2	2	15	6	2	2
sonar	5	15	2	2	5	6	4	3	4	1	3	15	6	2	2	3	2	5	3	4	4	2	15	15	13	15	15	5	1	1
wine	7	14	14	2	4	6	12	3	3	1	1	13	4	7	14	2	2	4	12	3	3	14	14	2	2	2	14	6	2	2
colon	5	7	2	2	5	7	6	5	3	1	1	8	2	2	2	8	2	4	8	8	2	2	8	7	2	8	8	4	1	1
prostate	4	11	2	2	2	4	10	3	3	1	6	10	2	2	2	2	2	4	3	2	3	2	11	2	2	2	11	3	3	3

Appendix B

Supplementary for Chapter 4

B.1 Proof of Remark 4.1

Proof. From the definition of BIC in (4.11), we can write the BIC for k -Means as,

$$BIC^{KM}(\Theta) = l(D; \Theta) - \frac{k(p+1)}{2} \log |X|. \quad (\text{B.1})$$

The number of parameters $|\Theta| = k(p+1)$ due to the kp cluster centers and k mixture proportions. The log likelihood function is defined as,

$$\begin{aligned} l(D; \Theta) &= \log \prod_{i=1}^n P(\mathbf{x}_i) \\ &= \sum_{i=1}^n \log \sum_{j=1}^k P(\mathbf{x}_i | \mathbf{x}_i \in C_j) P(\mathbf{x}_i \in C_j) \end{aligned}$$

Since all optimal C_j is obtained from k -Means clustering,

$$l(D; \Theta) = \sum_{i=1}^n [\log P(\mathbf{x}_i | \mathbf{x}_i \in C_j) + \log P(\mathbf{x}_i \in C_j)]$$

The maximum likelihood of the mixture proportions is

$$P(\mathbf{x}_i \in C_j)_{MLE} = \frac{|C_j|}{|X|},$$

and the sampling probability of \mathbf{x}_i from each C_j follows the normal distribution,

$$P(\mathbf{x}_i | \mathbf{x}_i \in C_j) = \frac{1}{(2\pi\sigma^2)^{p/2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2\right).$$

where all $\sigma_j = \sigma$ since in k -Means we assume all spherical clusters with equal variances. Then,

$$\begin{aligned}
 l(D; \Theta) &= \sum_{i=1}^n \left[\log \left(\frac{|C_j|}{|X|} \right) - \frac{p}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2 \right] \\
 &= \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \left[\log \left(\frac{|C_j|}{|X|} \right) - \frac{p}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2 \right] \\
 &= \sum_{j=1}^k \left[|C_j| \log \left(\frac{|C_j|}{|X|} \right) - \frac{p|C_j|}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2 \right].
 \end{aligned}$$

To estimate σ to maximize the log likelihood function, we equate the derivative to zero and solve for σ^2 ,

$$\begin{aligned}
 \frac{\partial}{\partial \sigma} l(D; \Theta) &= 0 \\
 \implies \sum_{j=1}^k \left[-\frac{p|C_j|}{\sigma} + \frac{1}{\sigma^3} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2 \right] &= 0 \\
 \implies \sigma^3 \left[-p|X|\sigma^2 + \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2 \right] &= 0 \\
 \implies \sigma^2 &= \frac{1}{p|X|} \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2.
 \end{aligned}$$

The unbiased estimator of the variance of each cluster σ_j^2 can then be written as,

$$\hat{\sigma}_{jUBE}^2 = \frac{1}{p(|C_j| - 1)} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2$$

Assuming all $\hat{\sigma}_j = \hat{\sigma}$, and summing over all clusters,

$$\begin{aligned}
 \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2 &= p \sum_{j=1}^k (|C_j| - 1) \hat{\sigma}_j^2 \\
 \implies \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2 &= p \hat{\sigma}^2 (|X| - k) \\
 \implies \hat{\sigma}^2 &= \frac{1}{p(|X| - k)} \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2. \tag{B.2}
 \end{aligned}$$

Substituting this estimator of σ^2 in the log likelihood function and simplifying, yields

$$l(D; \Theta) = \sum_{j=1}^k |C_j| \log |C_j| - |X| \log |X| - \frac{p|X|}{2} \log(2\pi\hat{\sigma}^2) - \frac{p}{2}(|X| - k).$$

Substituting in (B.1), we obtain,

$$\begin{aligned} BIC^{KM}(\Theta) = \sum_{j=1}^k |C_j| \log |C_j| - |X| \log |X| - \frac{p|X|}{2} \log(2\pi\hat{\sigma}^2) \\ - \frac{p}{2}(|X| - k) - \frac{k(p+1)}{2} \log |X|. \end{aligned} \tag{B.3}$$

Equations (B.3) and (B.2) prove Remark 4.1. □

B.2 Comparison of the execution times when using BIC in comparison to the Permutation Method

We randomly generated two-dimensional data sets containing 3 clusters at positions (0,0), (10,0) and (0,10) respectively. We generated 6 data sets of the following sizes: 2^{10} , 2^{11} , 2^{12} , 2^{13} , 2^{14} , and 2^{15} . Sparse k -Means using the Permutation Method (SKM+PM), and Sparse k -Means using the Bayesian Information Criterion (SKM+BIC) were run on each data set five times, and the average execution times were measured, which are shown in Table B.1. Similarly, Sparse Fuzzy c -Means using the Permutation Method (SFCM+PM), and Sparse Fuzzy c -Means using the Bayesian Information Criterion (SFCM+BIC) were run on each data set, and the execution times were measured, which are shown in Table B.2. The programs were executed on an Intel Core i7-4790 processor with base frequency of 3.60GHz, and with 32GB of 1600MHz DDR3 RAM.

Table B.1: Execution Times of SKM+PM and SKM+BIC.

Data Set Size (n, p)	Average Execution Time (in secs) of SKM+PM	Average Execution Time (in secs) of SKM+BIC
$(2^{10}, 2)$	252.8320	7.0758
$(2^{11}, 2)$	295.1284	7.8671
$(2^{12}, 2)$	369.8738	9.3612
$(2^{13}, 2)$	506.0423	12.1402
$(2^{14}, 2)$	771.5288	18.3846
$(2^{15}, 2)$	1293.2834	29.6807

Table B.2: Execution Times of SFCM+PM and SFCM+BIC.

Data Set Size (n, p)	Average Execution Time (in secs) of SFCM+PM	Average Execution Time (in secs) of SFCM+BIC
$(2^{10}, 2)$	251.9511	7.0849
$(2^{11}, 2)$	294.9346	7.8125
$(2^{12}, 2)$	368.8202	9.3230
$(2^{13}, 2)$	499.8744	12.0724
$(2^{14}, 2)$	764.4555	17.4731
$(2^{15}, 2)$	1276.1256	27.7616

B.3 Cluster Validity Indices for global sparse clustering model selection

In this section, we provide the definitions for the Cluster Validity Indices (CVIs) whose capability to perform sparse clustering model selection have been studied. The following notations are used throughout this section: $\mathbf{x}_1, \dots, \mathbf{x}_n$ are n data points, where $\mathbf{x}_i \in \mathbb{R}^d$. Clusters are represented as C_1, \dots, C_k , and the centers of each of the k clusters are $V = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$. Fuzzy memberships of \mathbf{x}_i to cluster C_j are represented by μ_{ij} , with m as the parameter for the degree of fuzziness. The following are the CVIs used in the comparative study.

1. Akaike Information Criterion (AIC) ([Akaike, 1974](#)):

$$AIC(V) = \sum_{j=1}^k \left\{ (-2(n_j \log(n_j) - (n_j - n) - \frac{n_j d}{2} \log(2\pi \Delta_j) - \frac{d(n_j - 1)}{2})) \right\} + 2k,$$

$$\text{where, } \Delta_j = \sum_{\mathbf{x}_i \in C_j} \frac{\|\mathbf{x}_i - \mathbf{v}_j\|^2}{d/(n-k)}.$$

2. Caliński and Harabasz (CH) Index ([Caliński and Harabasz, 1974](#)):

$$CH(V) = \frac{\text{trace } B}{\text{trace } W} \cdot \frac{n-k}{k-1},$$

where, B and W are between and within cluster scatter matrices:

$$\text{trace } B = \sum_{j=1}^k |C_j| \times \|\mathbf{v}_j - \bar{\mathbf{x}}\|^2, \text{ and,}$$

$$\text{trace } W = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2.$$

3. Classification Entropy (CE) ([Bezdek, 1975](#)):

$$CE(V) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c \mu_{ij} \log \mu_{ij}.$$

4. Davies-Bouldin (DB) Index ([Davies and Bouldin, 1979](#)):

$$DB(V) = \frac{1}{k} \sum_{j=1}^k \max_{j \neq l} \{DB_{jl}\},$$

$$\text{where, } DB_{jl} = \frac{\sqrt{\frac{\sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2}{|C_j|}} + \sqrt{\frac{\sum_{\mathbf{x}_i \in C_l} \|\mathbf{x}_i - \mathbf{v}_l\|^2}{|C_l|}}}{\|\mathbf{v}_j - \mathbf{v}_l\|}.$$

5. Fukuyama Sugeno (FS) Index ([Fukuyama and Sugeno, 1989](#))

$$FS(V) = \sum_{i=1}^n \sum_{j=1}^k \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2 - \sum_{j=1}^k \|\mathbf{v}_j - \bar{\mathbf{v}}\|^2,$$

$$\text{where, } \bar{\mathbf{v}} = \frac{1}{k} \sum_{j=1}^k \mathbf{v}_j$$

6. Fuzzy Hypervolume (FHV) (Dave, 1996):

$$FHV(V) = \sum_{j=1}^k \left(\frac{\sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2}{\sum_{i=1}^n \mu_{ij}^m} \right)^{1/2}.$$

7. I Index (Maulik and Bandyopadhyay, 2002):

$$I(V) = \left(\frac{1}{k} \times \frac{E_1}{E_k} \times D_k \right)^p,$$

$$\text{where, } E_k = \sum_{j=1}^k \sum_{i=1}^n \mu_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|,$$

$$\text{and, } D_k = \max_{i,j=1}^k \|\mathbf{v}_i - \mathbf{v}_j\|.$$

8. Modified Partition Coefficient (MPC) (Dave, 1996):

$$MPC(V) = 1 - \frac{k}{k-1} \left(1 - \sum_{i=1}^n \sum_{j=1}^k \frac{\mu_{ij}^2}{n} \right).$$

9. Partition Coefficient (PC) (Dave, 1996):

$$PC(V) = \sum_{i=1}^n \sum_{j=1}^k \frac{\mu_{ij}^2}{n}.$$

10. Partition Index (PI) (Bensaid et al., 1996):

$$PI(V) = \sum_{j=1}^k \frac{\sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2}{\sum_{i=1}^n \mu_{ij} \sum_{i=1}^k \|\mathbf{v}_j - \mathbf{v}_i\|^2}.$$

11. PBMF (Pakhira et al., 2004):

$$PBMF(V) = \left(\frac{1}{k} \times \frac{E_1}{J_m} \times D_k \right),$$

$$\text{where, } J_m(U, Z) = \sum_{i=1}^n \sum_{j=1}^k \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|,$$

$$\text{and, } E_1 = \sum_{i=1}^n \|\mathbf{x}_i - \bar{\mathbf{x}}\|,$$

$$\text{and, } D_k = \max_{i,j=1}^k \|\mathbf{v}_i - \mathbf{v}_j\|.$$

12. PCAES (Wu and Yang, 2005):

$$PCAES(V) = \sum_{j=1}^k \sum_{i=1}^n \mu_{ij}^2 / \mu_M - \sum_{j=1}^k \exp(-\min_{l \neq j} \{\|\mathbf{v}_j - \mathbf{v}_l\|^2\} / \beta_T),$$

$$\text{where, } \mu_M = \min_{1 \leq j \leq k} \left(\sum_{i=1}^n \mu_{ij}^2 \right),$$

$$\text{and, } \beta_T = \frac{\sum_{j=1}^k \|\mathbf{v}_j - \bar{\mathbf{v}}\|^2}{k}.$$

13. RLWY Index (Ren et al., 2016):

$$RLWY(V) = \sum_{j=1}^k \frac{\frac{1}{n_j} \sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2 + \frac{1}{k} \|\mathbf{v}_j - \bar{\mathbf{v}}\|^2}{\frac{1}{k-1} \sum_{l=1}^k \|\mathbf{v}_l - \mathbf{v}_j\|^2}$$

14. Rezaee Index (Rezaee, 2010):

$$Rezaee(V) = \frac{Sep(k)}{\max_k Sep(k)} + \frac{Comp(k)}{\max_k Comp(k)},$$

$$\text{and, } Sep(k) = \frac{2}{k(k-1)} \sum_{j \neq l}^k S_{rel}(C_j, C_l),$$

$$\text{and, } S(C_j, C_l) = \sum_{i=1}^n \min(\mu_{ij}, \mu_{il}) \times h(x_i),$$

$$\text{and, } h(x_i) = - \sum_{j=1}^k \mu_{ij} \log(\mu_{ij}).$$

15. Silhouette Index (Rousseeuw, 1987)

$$sil(k) = \frac{1}{n} \sum_{i=1}^n s_i,$$

where for all $\mathbf{x}_i \in C_j$,

$$a_i = d(\mathbf{x}_i, C_j), \text{ and } b_i = \min_{l \neq j} d(\mathbf{x}_i, C_l),$$

$$\text{and, } s_i = \begin{cases} \frac{b_i - a_i}{\max\{b_i, a_i\}} & , \text{ if } |C_j| > 1, \\ 1 & , \text{ if } |C_j| = 1 \end{cases}.$$

16. Xie-Beni (XB) Index (Xie and Beni, 1991):

$$XB(V) = \frac{\sum_{j=1}^c \sum_{i=1}^n \mu_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|^2}{n \min_{j,l} \|\mathbf{v}_j - \mathbf{v}_l\|^2}.$$

17. Xu Index (Xu, 1997):

$$Xu(V) = d \log \left(\sqrt{\frac{1}{n^2 d} \sum_{j=1}^c \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2} \right) + \log(k).$$

18. ZXF Index (Zhao et al., 2009b):

$$ZXF(V) = \frac{k * \sum_{j=1}^c \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{v}_j\|}{\sum_{j=1}^c |C_j| * \|\mathbf{v}_j - \bar{\mathbf{x}}\|}.$$

19. ZWZL Index (Zhang et al., 2008):

$$ZWZL(V) = \frac{Var(V)}{Sep(U)}$$

$$\text{where, } Var(V) = \left(\frac{c+1}{c-1}\right) \sum_{i=1}^n \sum_{j=1}^k \frac{1}{n_j} \mu_{ij} d^2(\mathbf{x}_i, \mathbf{v}_j),$$

$$d(\mathbf{x}, \mathbf{y}) = (1 - \exp(-\beta \|\mathbf{x} - \mathbf{y}\|^2))^{1/2}, \text{ and}$$

$$\beta = \left(\frac{\sum_{i=1}^n \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}{n}\right)^{-1},$$

$$\text{and } Sep(U) = 1 - \max_{i \neq j} [\max_{\mathbf{x}_k \in X} \min(\mu_{ki}, \mu_{kj})].$$

B.4 Experiment Results on comparison of approaches for sparse clustering models selection

In this section, we provide the average ARI achieved on synthetic data sets by the 21 approaches to sparse hard clustering model selection as well as sparse fuzzy clustering model selection. Tables B.3 to B.6 show the average ARI obtained by the approaches over the 44 synthetic data sets for Sparse k -Means. Tables B.7 to B.10 show the average ARI obtained by the approaches over the synthetic data sets for Sparse Fuzzy c -Means.

Table B.3: (Part I): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.

	10d- 40c- no0	10d- 40c- no1	10d- 40c- no2	10d- 40c- no3	10d- 40c- no4	10d- 40c- no5	10d- 40c- no6	10d- 40c- no7	10d- 40c- no8	10d- 40c- no9
AIC	0.2519	0.2425	0.2201	0.2088	0.2109	0.2222	0.2210	0.2231	0.2210	0.2428
CH	0.9627	0.9626	0.9629	0.9559	0.9600	0.9786	0.9795	0.9853	0.9591	0.9557
CE	0.9633	0.9626	0.9629	0.9559	0.9600	0.9651	0.9375	0.9853	0.9591	0.9613
DB	0.2519	0.2441	0.2250	0.2129	0.2253	0.2222	0.2210	0.2231	0.2210	0.2428
FS	0.9633	0.9626	0.9629	0.9559	0.9600	0.9651	0.9375	0.9853	0.9411	0.9613
FHV	0.9633	0.9626	0.9629	0.9253	0.9600	0.9651	0.9375	0.9853	0.9591	0.9613
I	0.9633	0.8896	0.9629	0.9559	0.8704	0.9651	0.9174	0.8659	0.9591	0.8972
MPC	0.9633	0.9626	0.9629	0.9559	0.9600	0.9786	0.9795	0.9853	0.9591	0.9613
PC	0.9633	0.9626	0.9629	0.9559	0.9600	0.9786	0.9795	0.9853	0.9591	0.9613
PI	0.9633	0.9626	0.9629	0.9253	0.9600	0.9651	0.9375	0.9853	0.9591	0.9613
PBMF	0.9633	0.8896	0.9629	0.9196	0.8704	0.9651	0.9174	0.8659	0.9591	0.8972
PCAES	0.2519	0.4254	0.2913	0.2129	0.2253	0.2222	0.2209	0.2231	0.3255	0.2428
RLWY	0.9633	0.9626	0.9629	0.9253	0.9600	0.9651	0.9375	0.9853	0.9591	0.9613
Rez	0.9633	0.9626	0.9629	0.9559	0.9600	0.9786	0.9375	0.9853	0.9411	0.9613
SIL	0.9627	0.9626	0.9629	0.9196	0.9600	0.9786	0.9795	0.9853	0.9591	0.9686
XB	0.9627	0.9291	0.6503	0.9559	0.8421	0.8944	0.6178	0.9236	0.5124	0.9613
Xu	0.9627	0.9626	0.9629	0.9559	0.9600	0.9786	0.9795	0.9853	0.9591	0.9557
ZXF	0.9633	0.9626	0.9629	0.9559	0.9600	0.9786	0.9795	0.9853	0.9591	0.9557
ZWZL	0.9504	0.2441	0.2201	0.2088	0.2253	0.9786	0.6178	0.9236	0.3255	0.3210
GAP	0.9295	0.8928	0.9308	0.8579	0.8947	0.9317	0.8873	0.9174	0.8959	0.8825
BIC	1.0000	0.9947	0.9784	0.9937	0.9952	0.9806	0.9817	0.9893	0.9688	0.9900

Table B.4: (Part II): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.

	10d- 20c- no0	10d- 20c- no1	10d- 20c- no2	10d- 20c- no3	10d- 20c- no4	10d- 20c- no5	10d- 20c- no6	10d- 20c- no7	10d- 20c- no8	10d- 20c- no9
AIC	0.3537	0.4140	0.2784	0.3180	0.3892	0.3171	0.3466	0.3309	0.3231	0.4325
CH	0.9963	0.9544	0.9886	0.9965	0.9987	0.9840	0.9564	0.9989	1.0000	1.0000
CE	0.9963	0.9544	0.9886	0.9965	0.9987	0.9840	0.9088	0.9989	1.0000	0.9373
DB	0.3537	0.4140	0.2784	0.3267	0.3300	0.3171	0.3466	0.3309	0.3231	0.4325
FS	0.9647	0.9544	0.8807	0.9965	0.9499	0.9257	0.9088	0.9508	1.0000	0.9373
FHV	0.9963	0.9544	0.9886	0.9965	0.9499	0.9257	0.9088	0.9989	1.0000	0.9373
I	0.9963	0.9544	0.8807	0.9965	0.9499	0.9257	0.9088	0.9989	1.0000	0.9373
MPC	0.9963	0.9733	0.9886	0.9965	0.9987	0.9840	0.9564	0.9989	1.0000	1.0000
PC	0.9963	0.9733	0.9886	0.9965	0.9987	0.9840	0.9564	0.9989	1.0000	1.0000
PI	0.9963	0.9733	0.9886	0.9965	0.9499	0.9257	0.9088	0.9989	1.0000	0.9373
PBMF	0.9945	0.8988	0.8807	0.9965	0.9499	0.9257	0.9088	0.9989	1.0000	0.9373
PCAES	0.4244	0.4140	0.3048	0.3180	0.3300	0.3171	0.5050	0.3133	0.3231	0.4256
RLWY	0.9963	0.9733	0.9886	0.9965	0.9499	0.9257	0.9088	0.9989	1.0000	0.9373
Rez	0.9963	0.9733	0.9886	0.9965	0.9987	0.9840	0.9088	0.9989	1.0000	0.9373
SIL	0.9963	0.9733	0.9886	0.9965	0.9987	0.9840	0.9564	0.9989	1.0000	1.0000
XB	0.9945	0.5642	0.9886	0.9949	0.9987	0.9840	0.7325	0.9989	0.9961	0.9932
Xu	0.9963	0.9544	0.9886	0.9965	0.9987	0.9840	0.9564	0.9989	1.0000	1.0000
ZXF	0.9963	0.9733	0.9886	0.9965	0.9987	0.9840	0.9088	0.9989	1.0000	0.9373
ZWZL	0.9945	0.9544	0.9886	0.9895	0.9987	0.9821	0.7325	0.9989	1.0000	1.0000
GAP	0.9963	0.9257	0.9592	0.9965	0.9667	0.9794	0.9639	0.9289	0.9237	0.9338
BIC	0.9963	1.0000	0.9980	0.9965	0.9987	1.0000	0.9952	1.0000	1.0000	1.0000

Table B.5: (Part III): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.

	10d- 10c- no0	10d- 10c- no1	10d- 10c- no2	10d- 10c- no3	10d- 10c- no4	10d- 10c- no5	10d- 10c- no6	10d- 10c- no7	10d- 10c- no8	10d- 10c- no9
AIC	0.3615	0.3034	0.3136	0.3226	0.3849	0.3566	0.3880	0.3036	0.2579	0.3241
CH	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
CE	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
DB	0.3615	0.3034	0.3269	0.3226	0.3849	0.3566	0.3880	0.3159	0.2459	0.3241
FS	0.4225	0.3173	0.5431	0.3226	0.3849	0.3566	0.3880	0.3036	0.2459	0.3241
FHV	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.6889	0.7552	0.7952	0.6477
I	0.8962	0.6485	0.6668	0.6500	0.6618	0.3516	0.4468	0.8418	0.5727	0.6307
MPC	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
PC	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
PI	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.6889	0.7552	0.7952	0.7507
PBMF	0.4875	0.6485	0.3881	0.6500	0.6618	0.3516	0.4468	0.8418	0.4656	0.3192
PCAES	0.3615	0.3255	0.3269	0.3226	0.3552	0.3566	0.3880	0.5888	0.2459	0.3241
RLWY	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.6889	0.7552	0.7952	0.7507
Rez	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
SIL	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
XB	0.7116	0.6485	0.8785	0.8031	0.9172	0.8467	0.7345	0.8418	0.7952	0.7507
Xu	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
ZXF	0.8962	0.8507	0.8785	0.8031	0.9172	0.9175	0.7345	0.8418	0.7952	0.7507
ZWZL	0.4225	0.6485	0.8785	0.4642	0.8670	0.8467	0.6889	0.8418	0.7952	0.3241
GAP	0.8451	0.8137	0.8792	0.8035	0.9164	0.9142	0.7129	0.7972	0.7977	0.8064
BIC	0.8962	0.8507	0.8785	0.8037	0.9172	0.9175	0.7346	0.8418	0.7952	0.7507

Table B.6: (Part IV): Average ARI achieved by the Sparse Model Selection approaches on Sparse k -Means Clusterings.

	10d- 4c- no0	10d- 4c- no1	10d- 4c- no2	10d- 4c- no3	10d- 4c- no4	10d- 4c- no5	10d- 4c- no6	10d- 4c- no7	10d- 4c- no8	10d- 4c- no9	2d- 4c- 2x2	2d- 6c- 3x2	2d- 8c- 4x2	2d- 15c- 5x3
AIC	0.7574	0.5483	0.3608	0.3582	0.4323	0.5474	0.2991	0.5307	0.5073	0.7427	1.0000	0.9761	0.9744	0.9548
CH	1.0000	0.9764	0.7879	0.5512	0.8085	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9772	0.9576
CE	1.0000	0.9764	0.7879	0.9277	0.8060	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
DB	0.7574	0.5483	0.3608	0.3562	0.4323	0.5666	0.2991	0.5307	0.5205	0.6904	1.0000	0.9761	0.9772	0.9576
FS	0.7631	0.8680	0.7879	0.5840	0.8085	0.9109	0.8043	0.8770	0.7895	0.7438	1.0000	0.9761	0.9744	0.9548
FHV	0.7631	0.9764	0.7879	0.5512	0.6743	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
I	1.0000	0.9537	0.3608	0.3604	0.8085	0.5671	0.9208	0.7310	0.7002	0.6904	1.0000	0.9761	0.9744	0.9548
MPC	1.0000	0.9764	0.7879	0.9277	0.8060	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
PC	1.0000	0.9764	0.7879	0.9277	0.8060	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
PI	1.0000	0.9764	0.7879	0.5258	0.6743	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
PBMF	0.8298	0.8680	0.3608	0.3604	0.8085	0.5671	0.2991	0.7310	0.7002	0.6904	1.0000	0.9761	0.9744	0.9548
PCAES	0.7574	0.5483	0.3608	0.3562	0.4323	0.5474	0.2991	0.5307	0.5073	0.6533	1.0000	0.9761	0.9772	0.9562
RLWY	1.0000	0.9764	0.7879	0.5258	0.6743	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
Rez	1.0000	0.9764	0.7879	0.9277	0.8060	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9562
SIL	1.0000	0.9764	0.7879	0.9277	0.8085	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9772	0.9562
XB	1.0000	0.9764	0.7879	0.9277	0.8085	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
Xu	1.0000	0.9764	0.7879	0.9277	0.8085	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9772	0.9576
ZXF	1.0000	0.9764	0.7879	0.9277	0.8060	0.9191	0.9468	0.9344	0.9547	0.9312	1.0000	0.9761	0.9744	0.9548
ZWZL	0.9894	0.9764	0.7879	0.9277	0.8085	0.9109	0.8577	0.9152	0.7895	0.9045	1.0000	0.9761	0.9772	0.9548
GAP	1.0000	0.9537	0.6958	0.4579	0.8042	0.8820	0.9208	0.6710	0.9377	0.9547	1.0000	0.9761	0.9744	0.9576
BIC	1.0000	1.0000	0.9761	0.9772	0.9627	0.9626	0.9629	0.9559	0.9600	0.9651	1.0000	0.9761	0.9772	0.9795

Table B.7: (Part I): Average ARI achieved by the Sparse Model Selection Approaches on Sparse Fuzzy c -Means Clusterings.

	10d- 40c- no0	10d- 40c- no1	10d- 40c- no2	10d- 40c- no3	10d- 40c- no4	10d- 40c- no5	10d- 40c- no6	10d- 40c- no7	10d- 40c- no8	10d- 40c- no9
AIC	0.2591	0.2338	0.2349	0.2145	0.2241	0.2328	0.2210	0.2188	0.2138	0.2453
CH	0.9390	0.9709	0.9544	0.9643	0.9464	0.9685	0.9282	0.9714	0.9701	0.9765
CE	0.9390	0.9185	0.9544	0.9278	0.9635	0.9399	0.9360	0.9714	0.9701	0.9665
DB	0.2591	0.2338	0.2349	0.2145	0.2241	0.2328	0.2210	0.2188	0.2138	0.2307
FS	0.8681	0.7723	0.9180	0.7425	0.9069	0.8636	0.9360	0.9592	0.9045	0.8423
FHV	0.8681	0.9185	0.9544	0.7425	0.9464	0.8636	0.9360	0.9592	0.9045	0.8423
I	0.9390	0.7723	0.9544	0.9559	0.9635	0.9685	0.9360	0.9592	0.9701	0.8423
MPC	0.9390	0.9709	0.9544	0.9278	0.9635	0.9685	0.9620	0.9714	0.9701	0.9665
PC	0.9390	0.9709	0.9544	0.9278	0.9635	0.9685	0.9620	0.9714	0.9701	0.9665
PI	0.9390	0.9185	0.9544	0.9278	0.9635	0.9399	0.9360	0.9592	0.9701	0.9665
PBMF	0.9390	0.7723	0.9544	0.9559	0.9635	0.9685	0.9360	0.9592	0.9701	0.8423
PCAES	0.2591	0.4565	0.4116	0.2115	0.2161	0.2328	0.2210	0.2188	0.3132	0.2307
RLWY	0.9390	0.9185	0.9544	0.9278	0.9635	0.9399	0.9360	0.9592	0.9701	0.9665
Rez	0.9390	0.9185	0.9544	0.9278	0.9635	0.9685	0.9360	0.9714	0.9701	0.9665
SIL	0.9410	0.9709	0.9544	0.9643	0.9635	0.9685	0.9620	0.9714	0.9701	0.9765
XB	0.9390	0.9709	0.4116	0.9278	0.9464	0.4934	0.7028	0.8724	0.8295	0.8423
Xu	0.9410	0.9709	0.9544	0.9643	0.9464	0.9685	0.9282	0.9714	0.9701	0.9765
ZXF	0.9390	0.9185	0.9544	0.9278	0.9635	0.9685	0.9360	0.9714	0.9701	0.9665
ZWZL	0.9065	0.9709	0.9459	0.9278	0.9464	0.9685	0.9282	0.9714	0.9701	0.9765
GAP	0.9929	0.9571	0.9377	0.9500	0.9510	0.9777	0.9634	0.9587	0.9678	0.9624
BIC	1.0000	0.9744	0.9949	0.9932	0.9937	0.9606	0.9797	0.9946	0.9702	0.9842

Table B.8: (Part II): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.

	10d- 20c- no0	10d- 20c- no1	10d- 20c- no2	10d- 20c- no3	10d- 20c- no4	10d- 20c- no5	10d- 20c- no6	10d- 20c- no7	10d- 20c- no8	10d- 20c- no9
AIC	0.4094	0.4171	0.2920	0.3311	0.3774	0.3187	0.3383	0.3266	0.3335	0.4425
CH	0.9963	0.9544	0.9847	0.9965	0.9980	0.9783	0.9910	1.0000	1.0000	0.9920
CE	0.9963	0.9544	0.9707	0.9965	0.9980	0.9756	0.9910	1.0000	1.0000	0.9311
DB	0.4146	0.4171	0.2794	0.3311	0.3774	0.3187	0.3383	0.3266	0.3335	0.4425
FS	0.9163	0.8740	0.9249	0.9392	0.9511	0.9075	0.8991	0.9224	0.9952	0.9632
FHV	0.9163	0.8740	0.9249	0.9392	0.9511	0.9075	0.8745	0.9224	0.9952	0.9632
I	0.9163	0.9544	0.9249	0.9965	0.9511	0.9536	0.9367	1.0000	1.0000	0.9311
MPC	0.9963	0.9851	0.9847	0.9965	0.9980	0.9756	0.9910	1.0000	1.0000	0.9920
PC	0.9963	0.9851	0.9847	0.9965	0.9980	0.9756	0.9910	1.0000	1.0000	0.9920
PI	0.9163	0.8740	0.9249	0.9965	0.9511	0.9756	0.9367	1.0000	1.0000	0.9311
PBMF	0.9163	0.9231	0.9249	0.9965	0.8155	0.9536	0.8075	0.9224	1.0000	0.9091
PCAES	0.4094	0.4171	0.2794	0.3466	0.3774	0.3124	0.4335	0.3372	0.3469	0.4425
RLWY	0.9163	0.8740	0.9249	0.9965	0.9511	0.9756	0.9367	1.0000	1.0000	0.9311
Rez	0.9963	0.9544	0.9707	0.9965	0.9980	0.9756	0.9910	1.0000	1.0000	0.9920
SIL	0.9963	0.9851	0.9847	0.9965	0.9980	0.9783	0.9910	1.0000	1.0000	0.9920
XB	0.9963	0.8920	0.9847	0.9965	0.9980	0.9783	0.9910	1.0000	1.0000	0.9920
Xu	0.9963	0.9528	0.9847	0.9965	0.9980	0.9783	0.9910	1.0000	1.0000	0.9920
ZXF	0.9963	0.9544	0.9847	0.9965	0.9980	0.9756	0.9910	1.0000	1.0000	0.9920
ZWZL	0.9577	0.9528	0.9847	0.9965	0.9980	0.9783	0.9910	1.0000	0.9952	0.9920
GAP	0.9715	0.9481	0.9980	0.9965	0.9987	1.0000	0.9910	1.0000	1.0000	1.0000
BIC	0.9963	1.0000	0.9980	0.9965	0.9987	1.0000	0.9910	1.0000	1.0000	1.0000

Table B.9: (Part III): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.

	10d- 10c- no0	10d- 10c- no1	10d- 10c- no2	10d- 10c- no3	10d- 10c- no4	10d- 10c- no5	10d- 10c- no6	10d- 10c- no7	10d- 10c- no8	10d- 10c- no9
AIC	0.4074	0.3016	0.3419	0.2947	0.3925	0.2796	0.4056	0.2711	0.2181	0.3129
CH	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
CE	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
DB	0.3579	0.3016	0.3419	0.3222	0.3610	0.2720	0.4056	0.2711	0.2181	0.3962
FS	0.4074	0.3138	0.5212	0.7719	0.3925	0.2796	0.6828	0.6674	0.6568	0.3962
FHV	0.5977	0.8052	0.5212	0.7719	0.3925	0.9023	0.6828	0.7223	0.6568	0.3962
I	0.8064	0.7647	0.6544	0.4962	0.6483	0.9023	0.4813	0.7870	0.7587	0.5956
MPC	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
PC	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
PI	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
PBMF	0.5145	0.6523	0.3887	0.4962	0.4665	0.9023	0.4813	0.7870	0.5707	0.3077
PCAES	0.3579	0.3138	0.3419	0.2947	0.3610	0.3246	0.4813	0.5265	0.5707	0.3962
RLWY	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
Rez	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
SIL	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
XB	0.6961	0.7647	0.8623	0.7719	0.9082	0.8347	0.7046	0.6251	0.7031	0.3678
Xu	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
ZXF	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7870	0.7587	0.6691
ZWZL	0.8064	0.8052	0.8623	0.7719	0.9082	0.9023	0.6828	0.7870	0.7587	0.6691
GAP	0.8420	0.8366	0.8623	0.7646	0.9082	0.9023	0.7010	0.8243	0.7670	0.7384
BIC	0.8490	0.8052	0.8623	0.7719	0.9082	0.9023	0.7740	0.7876	0.7670	0.6731

Table B.10: (Part IV): Average ARI achieved by the Sparse Model Selection approaches on Sparse Fuzzy c -Means Clusterings.

	10d- 4c- no0	10d- 4c- no1	10d- 4c- no2	10d- 4c- no3	10d- 4c- no4	10d- 4c- no5	10d- 4c- no6	10d- 4c- no7	10d- 4c- no8	10d- 4c- no9	2d- 4c- 2x2	2d- 6c- 3x2	2d- 8c- 4x2	2d- 15c- 5x3
AIC	0.7577	0.5609	0.3646	0.3498	0.4357	0.5472	0.5058	0.5464	0.5007	0.7398	1.0000	0.9762	0.9744	0.9548
CH	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9562
CE	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
DB	0.7577	0.5609	0.3646	0.3587	0.4470	0.5656	0.5058	0.5464	0.4910	0.6586	1.0000	0.9762	0.9744	0.9576
FS	0.7577	0.8517	0.6945	0.5270	0.7656	0.8777	0.8657	0.9080	0.8613	0.7507	1.0000	0.9761	0.9744	0.9576
FHV	0.7577	0.8517	0.6945	0.5270	0.7656	0.8777	0.8291	0.9080	0.7614	0.9547	1.0000	0.9761	0.9744	0.9576
I	1.0000	0.9192	0.4239	0.3569	0.7656	0.5655	0.9317	0.7276	0.6731	0.6859	1.0000	0.9762	0.9744	0.9548
MPC	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
PC	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
PI	1.0000	0.9703	0.5608	0.5018	0.6732	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
PBMF	0.8957	0.7024	0.4239	0.3569	0.7656	0.5655	0.9317	0.7276	0.6731	0.6859	1.0000	0.9762	0.9744	0.9548
PCAES	0.7577	0.5609	0.5171	0.3498	0.4470	0.5472	0.5304	0.5464	0.9251	0.6586	1.0000	0.9761	0.9744	0.9548
RLWY	1.0000	0.9703	0.5608	0.5018	0.6732	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
Rez	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
SIL	1.0000	0.9703	0.6945	0.5270	0.8036	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
XB	1.0000	0.9703	0.6945	0.4336	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
Xu	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
ZXF	1.0000	0.9703	0.6945	0.5270	0.8013	0.9191	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9576
ZWZL	0.9994	0.8517	0.6871	0.5018	0.6633	0.9198	0.9358	0.9080	0.9455	0.9535	1.0000	0.9761	0.9744	0.9548
GAP	0.9994	0.9703	0.6871	0.5222	0.6864	0.9198	0.9358	0.9239	0.9455	0.9547	1.0000	0.9761	0.9744	0.9548
BIC	1.0000	1.0000	0.9761	0.9744	0.9390	0.9709	0.9544	0.9559	0.9635	0.9685	1.0000	0.9761	0.9744	0.9576

B.5 Experiment Results of sparse clustering methods on synthetic data sets

Table B.11 shows the average ARI achieved on the synthetic data sets by the proposed Sparse k -Means using BIC (SKM+BIC) method, in contention with the methods of Sparse k -Means using the Permutation Method (SKM+PM), Robust Sparse k -Means (RSKM) (Kondo et al., 2012) and Structured Sparse k -Means (SSKM) (Gong et al., 2018).

Table B.11: Average ARI achieved on Synthetic Datasets by Sparse Hard Clustering Methods.

Dataset	SKM+GAP	RSKM	SSKM	SKM+BIC
10d-40c-no0	0.9295	0.9190	0.8286	1.0000
10d-40c-no1	0.8928	0.9386	0.9093	0.9947
10d-40c-no2	0.9308	0.9297	0.8980	0.9784
10d-40c-no3	0.8579	0.8503	0.2966	0.9937
10d-40c-no4	0.8947	0.9102	0.8007	0.9952
10d-40c-no5	0.9317	0.8611	0.7012	0.9806
10d-40c-no6	0.8873	0.8906	0.4964	0.9817
10d-40c-no7	0.9174	0.9010	0.8005	0.9893
10d-40c-no8	0.8959	0.9551	0.8969	0.9688
10d-40c-no9	0.8825	0.9266	0.8055	0.9900
10d-20c-no0	0.9963	0.9371	0.8865	0.9963
10d-20c-no1	0.9257	0.9299	0.8087	1.0000
10d-20c-no2	0.9592	0.8749	0.5793	0.9980
10d-20c-no3	0.9965	0.9340	0.361	0.9965
10d-20c-no4	0.9667	0.8810	0.8684	0.9987
10d-20c-no5	0.9794	0.9323	0.8482	1.0000
10d-20c-no6	0.9639	0.9288	0.4730	0.9952
10d-20c-no7	0.9289	0.8932	0.8634	1.0000
10d-20c-no8	0.9237	0.8894	0.8384	1.0000
10d-20c-no9	0.9338	0.9385	0.8882	1.0000
10d-10c-no0	0.8451	0.7333	0.1190	0.8962
10d-10c-no1	0.8137	0.8349	0.1503	0.8507
10d-10c-no2	0.8792	0.7723	0.0816	0.8785
10d-10c-no3	0.8035	0.7991	0.0941	0.8037
10d-10c-no4	0.9164	0.7552	0.372	0.9172
10d-10c-no5	0.9142	0.9289	0.2481	0.9175
10d-10c-no6	0.7129	0.7768	0.2262	0.7346
10d-10c-no7	0.7972	0.6884	0.3764	0.8418
10d-10c-no8	0.7977	0.7668	0.1826	0.7952
10d-10c-no9	0.8064	0.7642	0.1944	0.7507
10d-4c-no0	1.0000	0.9994	0.7326	1.0000
10d-4c-no1	0.9537	0.9616	0.4584	1.0000
10d-4c-no2	0.6958	0.8573	0.3485	0.9761
10d-4c-no3	0.4579	0.9614	0.4095	0.9772
10d-4c-no4	0.8042	0.8336	0.3580	0.9627
10d-4c-no5	0.8820	0.3666	0.2834	0.9626
10d-4c-no6	0.9208	0.9541	0.1825	0.9629
10d-4c-no7	0.6710	0.9671	0.2941	0.9559
10d-4c-no8	0.9377	0.9850	0.0324	0.9600
10d-4c-no9	0.9547	0.6628	0.3334	0.9651
2d-4c-2x2	1.0000	0.3508	0.4969	1.0000
2d-6c-3x2	0.9761	0.9761	0.4412	0.9761
2d-8c-4x2	0.9744	0.4589	0.4528	0.9772
2d-15c-5x3	0.9576	0.8649	0.3453	0.9795
Avg. Ranks	2.25	2.48	3.93	1.18
Hyp Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
pval	4.38E-07	1.50E-07	7.62E-09	-

Table B.12 shows the average ARI achieved on the synthetic data sets by the proposed Sparse Fuzzy c -Means using BIC (SFCM+BIC) method, in contention with the methods

of Sparse Fuzzy c -Means using the Permutation Method (SFCM+PM), and Robust Sparse Fuzzy c -Means using the $\ell_{2,1}$ norm (RSFCM($\ell_{2,1}$)) and the ℓ_1 -capped norm (RSFCM(ℓ_1 -c)) (Xu et al., 2016).

Table B.12: Average ARI achieved on Synthetic Datasets by Sparse Fuzzy Clustering Methods.

Dataset	SFCM+GAP	RSFCM-l21	RSFCM-l1c	SFCM+BIC
10d-10c-no0	0.8420	0.6349	0.6894	0.8490
10d-10c-no1	0.8366	0.4929	0.4913	0.8052
10d-10c-no2	0.8623	0.3501	0.3456	0.8623
10d-10c-no3	0.7646	0.4959	0.4799	0.7719
10d-10c-no4	0.9082	0.5307	0.5162	0.9082
10d-10c-no5	0.9023	0.3718	0.3969	0.9023
10d-10c-no6	0.7010	0.4214	0.4986	0.7740
10d-10c-no7	0.8243	0.3635	0.3524	0.7876
10d-10c-no8	0.7670	0.3791	0.3348	0.7670
10d-10c-no9	0.7384	0.4582	0.4136	0.6731
10d-20c-no0	0.9715	0.7310	0.7420	0.9963
10d-20c-no1	0.9481	0.7571	0.7290	1.0000
10d-20c-no2	0.9980	0.7201	0.6435	0.9980
10d-20c-no3	0.9965	0.7778	0.6247	0.9965
10d-20c-no4	0.9987	0.7928	0.6803	0.9987
10d-20c-no5	1.0000	0.8149	0.7053	1.0000
10d-20c-no6	0.9910	0.8335	0.6744	0.9910
10d-20c-no7	1.0000	0.7305	0.7234	1.0000
10d-20c-no8	1.0000	0.7748	0.7507	1.0000
10d-20c-no9	1.0000	0.8175	0.6584	1.0000
10d-40c-no0	0.9929	0.6961	0.6333	1.0000
10d-40c-no1	0.9571	0.6990	0.5855	0.9744
10d-40c-no2	0.9377	0.6975	0.6735	0.9949
10d-40c-no3	0.9500	0.4371	0.4207	0.9932
10d-40c-no4	0.9510	0.5974	0.5272	0.9937
10d-40c-no5	0.9777	0.5777	0.5233	0.9606
10d-40c-no6	0.9634	0.6773	0.6553	0.9797
10d-40c-no7	0.9587	0.5826	0.5237	0.9946
10d-40c-no8	0.9678	0.5064	0.5614	0.9702
10d-40c-no9	0.9624	0.6825	0.6920	0.9842
10d-4c-no0	0.9994	0.7638	0.7307	1.0000
10d-4c-no1	0.9703	0.7768	0.7485	1.0000
10d-4c-no2	0.6871	0.7560	0.7227	0.9761
10d-4c-no3	0.5222	0.7297	0.7057	0.9744
10d-4c-no4	0.6864	0.7732	0.7363	0.9390
10d-4c-no5	0.9198	0.3872	0.3010	0.9709
10d-4c-no6	0.9358	0.3573	0.3640	0.9544
10d-4c-no7	0.9239	0.8859	0.8338	0.9559
10d-4c-no8	0.9455	0.6244	0.8407	0.9635
10d-4c-no9	0.9547	0.8858	0.8446	0.9685
2d-4c-2x2	1.0000	0.9137	0.8368	1.0000
2d-6c-3x2	0.9761	0.9601	0.6122	0.9761
2d-8c-4x2	0.9744	0.9744	0.7057	0.9744
2d-15-5x3	0.9548	0.7766	0.6599	0.9576
Avg. Ranks	1.70	3.07	3.75	1.09
Hyp Test	\mathcal{H}_1	\mathcal{H}_1	\mathcal{H}_1	-
pval	1.32E-03	1.12E-08	7.61E-09	-

B.6 Influence of the sparsity of the dataset

An interesting study can be the influence of the sparsity of the dataset on the sparse center-based clustering methods, when the proposed BIC based approach is used in comparison to the permutation method based approach using the GAP statistic. We consider two datasets from the collection of ten-dimensional datasets from the MOCK collection (Handl and Knowles, 2007). One of the datasets contains ten clusters, whereas the other contains forty clusters. The TSNE plots of the datasets allows us to visualize the dataset under consideration. To this dataset we add an increasing number of noisy features, from two to twenty in increments of two. The noisy features that are added each are based on a random one-dimensional Gaussian with zero mean and variance equal to average feature variance of the dataset. On these two datasets, we first study the performance of SKM+GAP and SKM+BIC as the number of noisy features is increased, following which we similarly study the clustering performance of SFCM+GAP in comparison to SFCM+BIC. The performance of the sparse clustering methods are measured in terms of the average ARI achieved on the dataset over 20 runs of each method. The experiment protocol of Section 4.4.1 is followed.

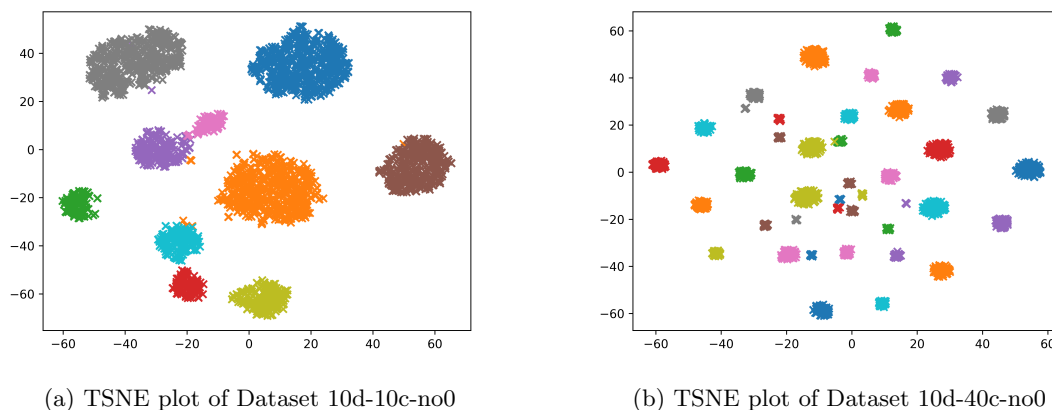


Figure B.1: TSNE plots of two 10-dimensional datasets from the MOCK collection of datasets (Handl and Knowles, 2007). To these datasets, noise features are added with variance comparable to the average feature variance of the datasets.

Table B.13 shows the average ARI achieved by SKM+GAP in comparison to SKM+BIC. We observe that for both the datasets, SKM+BIC generally outperforms SKM+GAP across different number of noisy features. For the cases where SKM+GAP attains the highest average ARI, SKM+BIC also performs competitively. Thus we observe a significantly more robust clustering performance by the proposed SKM+BIC compared to that of SKM+GAP. Similarly, in Table B.14 we observe the clustering performances of SFCM+GAP in comparison to the proposed SFCM+BIC. From the results we observe that across different number of noisy features, SFCM+BIC performs competitively compared to SFCM+GAP, outperforming the latter in more cases for the dataset with ten clusters. Therefore we can conclude that SFCM+BIC provides a low-cost alternative that performs as well as SFCM+GAP, while SKM+BIC is a better choice for the sparse clustering of a dataset in comparison to SKM+GAP while also incurring significantly lower computation costs.

Table B.13: Average ARI achieved by SKM+GAP and SKM+BIC in the presence of increasing number of noise features.

Dataset	#noise features	SKM+GAP	SKM+BIC
10d-10c-no0	2	0.8262	0.8635
	4	0.8657	0.8640
	6	0.8629	0.8640
	8	0.8413	0.9018
	10	0.9036	0.9038
	12	0.8629	0.8629
	14	0.8150	0.9030
	16	0.9018	0.9022
	18	0.8598	0.9027
	20	0.8118	0.8635
10d-40c-no0	2	0.9161	0.9440
	4	0.8725	0.9227
	6	0.9057	0.9103
	8	0.8915	0.9384
	10	0.9190	0.9447
	12	0.8775	0.9281
	14	0.9430	0.9232
	16	0.8863	0.9436
	18	0.9085	0.9390
	20	0.9397	0.9417

Table B.14: Average ARI achieved by SFCM+GAP and SFCM+BIC in the presence of increasing number of noise features.

Dataset	#noise features	SFCM+GAP	SFCM+BIC
10d-10c-no0	2	0.8523	0.8387
	4	0.6696	0.8199
	6	0.7833	0.7416
	8	0.6702	0.7762
	10	0.6799	0.7540
	12	0.6296	0.7733
	14	0.6186	0.6339
	16	0.6663	0.7354
	18	0.6623	0.6267
	20	0.6278	0.6143
10d-40c-no0	2	0.8839	0.9054
	4	0.8645	0.8827
	6	0.8864	0.8962
	8	0.8874	0.8421
	10	0.8808	0.8455
	12	0.8848	0.8719
	14	0.8823	0.8864
	16	0.8868	0.8882
	18	0.8420	0.8616
	20	0.8980	0.8397

Appendix C

Supplementary for Chapter 5

C.1 Tuning of parameters for ECM

In this section we study the changes in the maximum Adjusted Rand Index (ARI) for different values of the parameters of the multi-objective optimization methods used. NSGA-II has the following parameters:

1. *pop*: Size of the population.
2. *FE*: Number of fitness evaluations.
3. *pool*: Fraction of the population undergoing genetic operations.
4. *tour*: During tournament selection, the number of solutions from which one is selected.
5. *mu*: Distribution index for crossover.
6. *mum*: Distribution index for mutation.

MOEA/D has the following parameters:

1. *pop*: Size of the population.
2. *T*: Size of the neighbourhood.
3. *FE*: Number of fitness evaluations.
4. *F*: Parameter for mutation in Differential Evolution.
5. *Cr*: Parameter for crossover in Differential Evolution.

The effects of tuning the parameter values are observed over three datasets containing three clusters that are well-separated, slightly overlapped, or highly overlapped. The datasets are shown in Fig. C.1.

In our experiments, we ran ECM-NSGA-II with the following parameter values: $pop = 50$, $FE = 5000$, $pool = 0.5$, $tour = 2$, $mu = 20$, and $mum = 20$. In Figs. C.2 to C.7, we observe the variations in the best ARI achieved for different values of each parameter. When varying a single parameter, the other parameters are set to the values we used in our experiments.

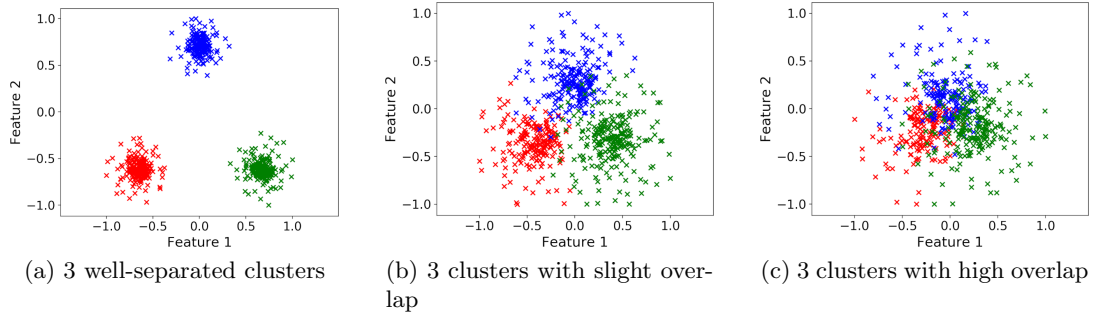
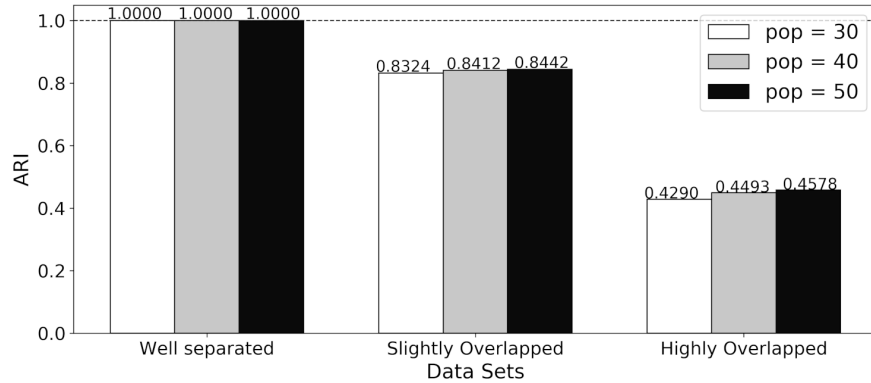


Figure C.1: Data set containing 3 clusters with different levels of overlap

In each figure, one can observe that the variation in best ARI is very small, suggesting that the methods are quite resilient to the choice of parameters. For the parameters pop , FE , $pool$, μ and mum , the best ARI achieved increases with an increase in the parameter values considered in the experiments. For $tour$, the maximum ARI decreases with the increase in parameter values.

Similarly, in our experiments, ECM-MOEA/D was executed with the following parameters values: $pop = 50$, $FE = 5000$, $T = 50$, $F = 0.5$, $Cr = 0.5$. Figs. C.8 to C.12 show small variations in the best ARI achieved with variations in each parameter, while keeping all other parameters fixed. This indicates that the methods are quite resilient to the choice of parameter values. For pop , FE and T , the best ARI achieved increases with an increase in the parameter values considered in the experiments. For F and Cr , the best ARI is respectively achieved when $F = 0.5$ and $Cr = 0.5$.

Figure C.2: The variation in maximum ARI with variation in the population size (pop) for ECM-NSGA-II

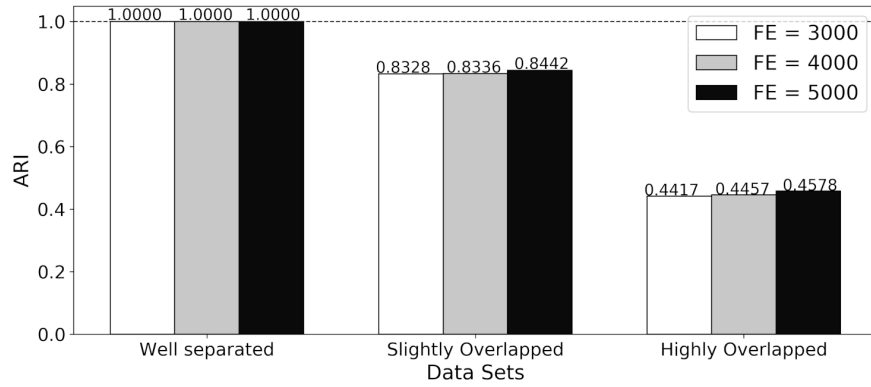


Figure C.3: The variation in maximum ARI with variation in the number of fitness evaluations (FE) for ECM-NSGA-II

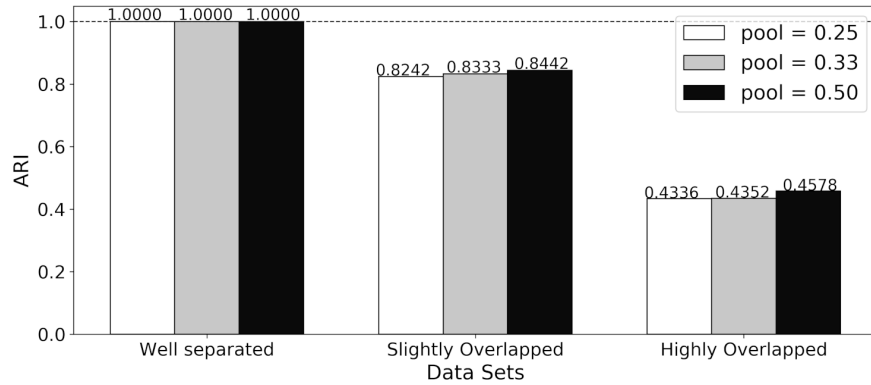


Figure C.4: The variation in maximum ARI with variation in the fraction of population undergoing genetic operation ($pool$) for ECM-NSGA-II

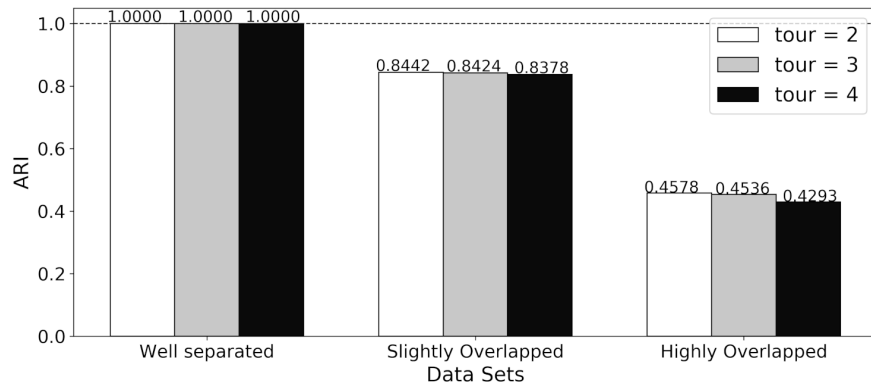


Figure C.5: The variation in maximum ARI with variation in the number of solutions ($tour$) from which one solution is selected during tournament selection for ECM-NSGA-II

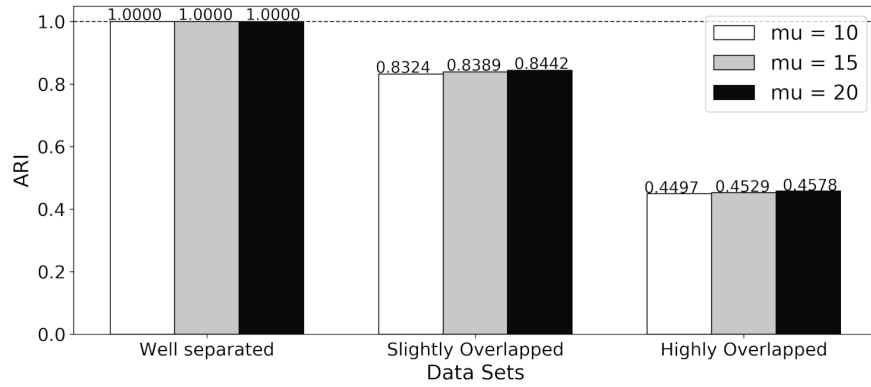


Figure C.6: The variation in maximum ARI with variation in the distribution index for crossover (μ) for ECM-NSGA-II

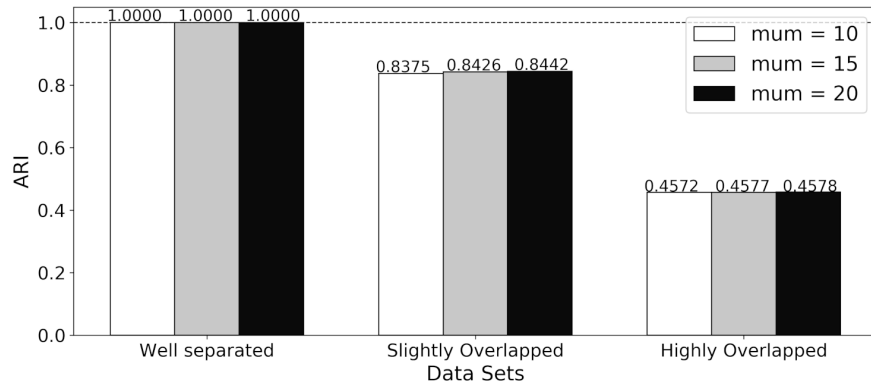


Figure C.7: The variation in maximum ARI with variation in the distribution index for mutation (mum) for ECM-NSGA-II

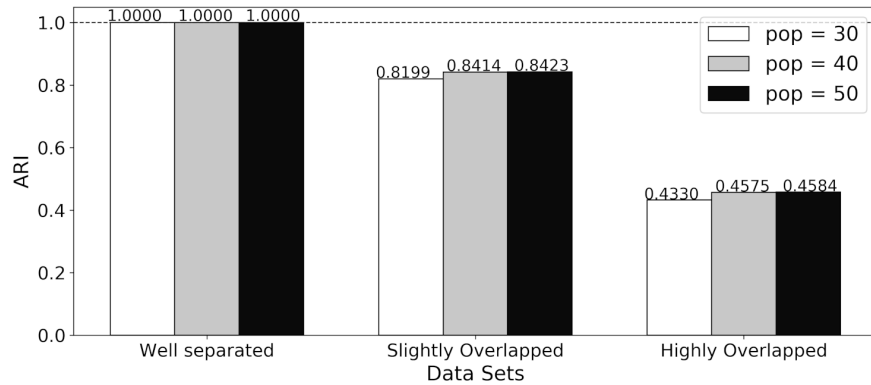


Figure C.8: The variation in maximum ARI with variation in the population size (pop) for ECM-MOEA/D

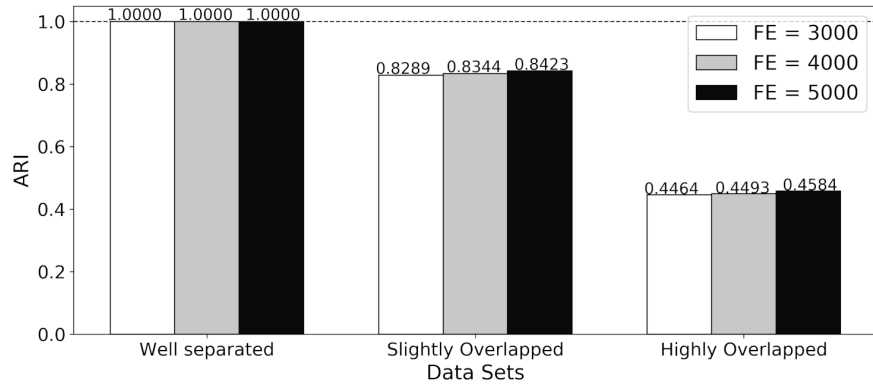


Figure C.9: The variation in maximum ARI with variation in the number of fitness evaluations (FE) for ECM-MOEA/D

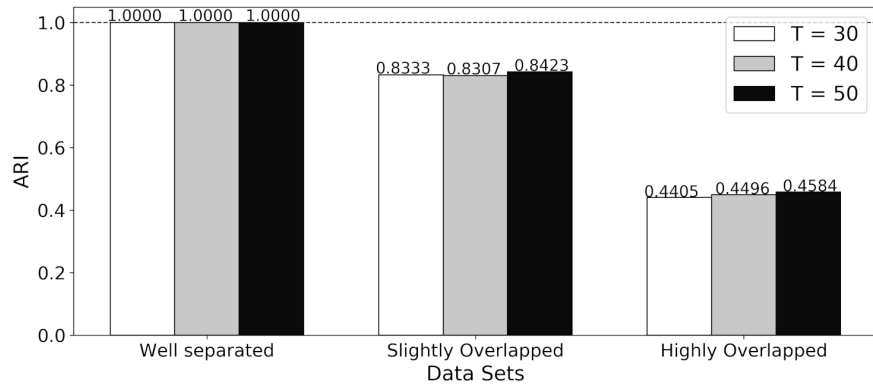


Figure C.10: The variation in maximum ARI with variation in the neighbourhood size (T) for ECM-MOEA/D

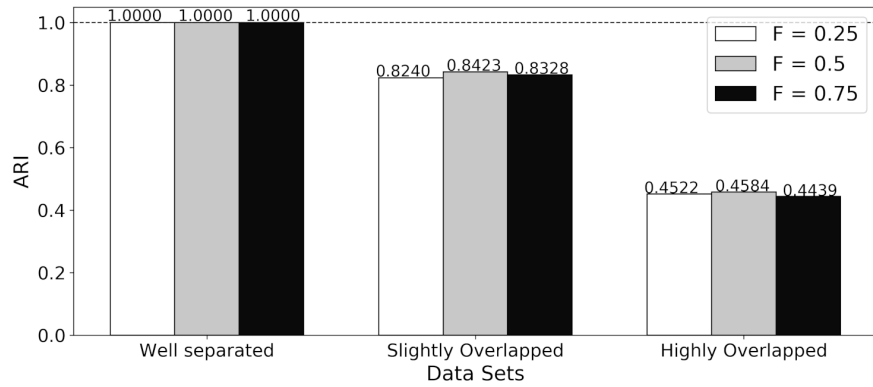


Figure C.11: The variation in maximum ARI with variation in the mutation parameter F in Differential Evolution for ECM-MOEA/D

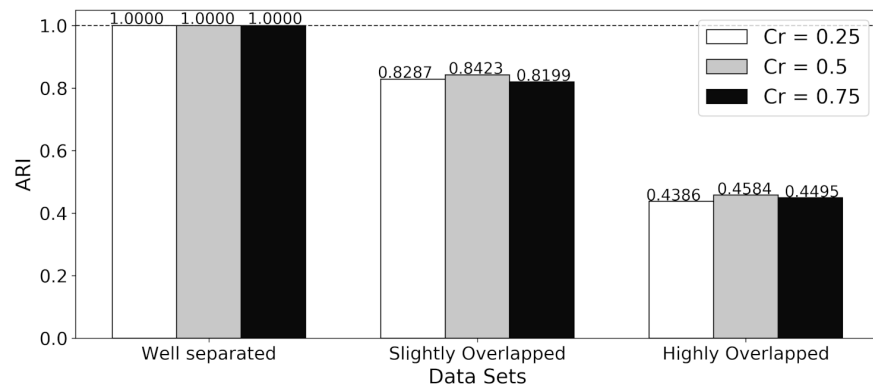


Figure C.12: The variation in maximum ARI with variation in crossover parameter Cr in Differential Evolution for ECM-MOEA/D

C.2 ECM on datasets with different levels of overlap

In this section, we demonstrate the workings of ECM on three datasets, each of which has three clusters with different levels of overlap between them. The three datasets we consider are shown in Figure C.13. The leftmost dataset *data1-well-separated* contains three well-separated clusters which should be identified by a fuzzy clustering method with low levels of fuzziness. The dataset shown in the middle *data2-slight-overlaps* contains three clusters with slight overlap between the clusters. A corresponding fuzzy clustering model should identify clusters with a level of fuzziness that is best suited to the underlying degree of overlap. The rightmost dataset *data3-high-overlaps* contains three clusters that are highly overlapped. A fuzzy clustering method should identify the clusters with a higher level of fuzziness to closely match the underlying degree of overlap between the clusters.

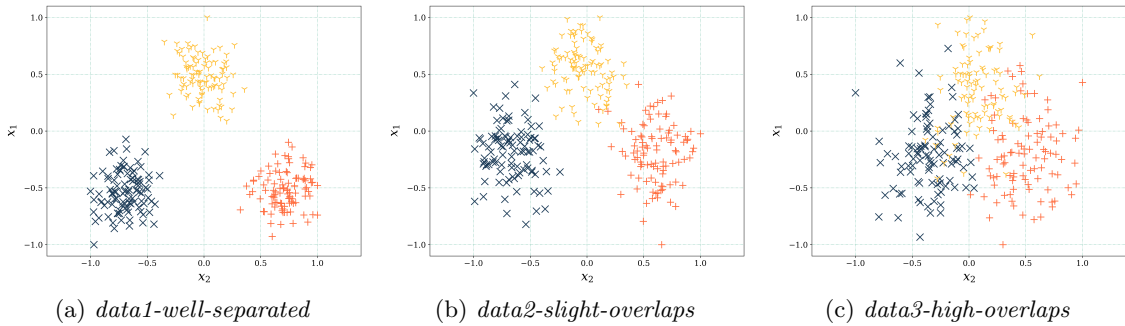


Figure C.13: We consider three datasets to demonstrate the working of the proposed ECM, shown from left to right named respectively as *data1-well-separated*, *data2-slight-overlaps*, and *data3-high-overlaps*.

When ECM-NSGA-II and ECM-MOEA/D are run on *data1-well-separated*, we obtain a wide Pareto front of clustering solutions as shown in Figures C.14 and C.15. The best clustering solution for these datasets can be identified as the leftmost solution on the Pareto front, where the most discrete solutions have been identified by ECM corresponding to the lowest degree of fuzziness among all the solutions on the Pareto front. This behaviour can be observed for both ECM-NSGA-II and ECM-MOEA/D. One can traverse the rest of the solutions on the Pareto front to observe the rest of the clustering solutions that have been identified, which correspond to clusterings with increasing levels of fuzziness until all clusters are overlapped in the rightmost solution of the Pareto front.

On the dataset *data2-slight-overlaps* we observe that ECM-NSGA-II and ECM-MOEA/D successfully return a wide Pareto front of clustering solutions in Figures C.16 and C.17. One can observe that the leftmost solution on the Pareto front corresponds to a clustering with the lowest levels of fuzziness; the rightmost solution results in a clustering with the highest levels of fuzziness where all clusters are overlapped, and between these two extreme cases we obtain a wide range of clustering solutions. Among these solutions we can obtain a clustering with a level of fuzziness that is best suited to the low levels of overlap between the clusters.

On the dataset *data3-high-overlaps*, the Pareto fronts identified by ECM-NSGA-II and ECM-MOEA/D are shown in Figures C.18 and C.19. From the Pareto front we can obtain a clustering solution with high levels of fuzziness to match the high degree of overlap present

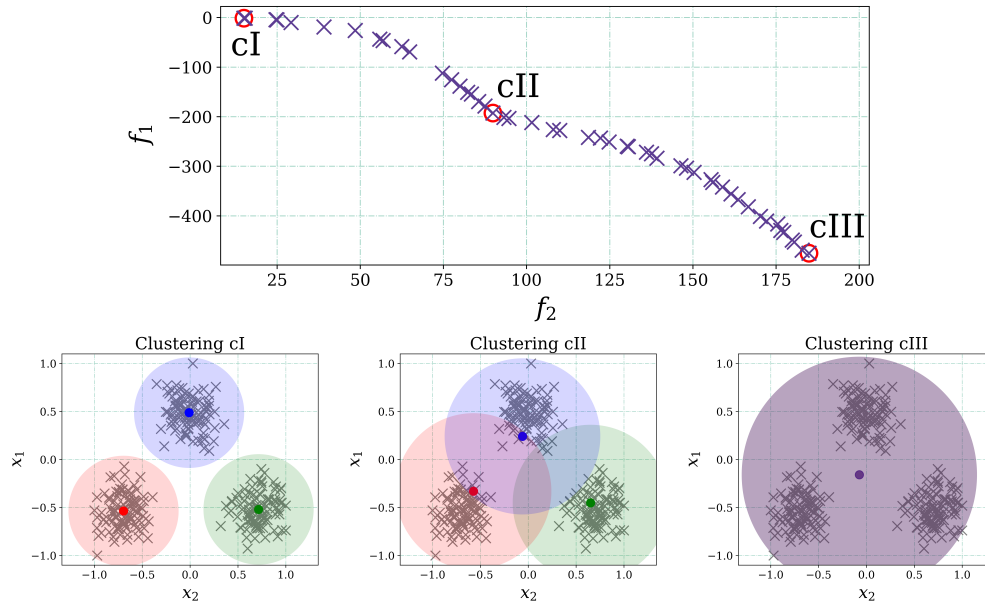


Figure C.14: For the dataset *data1-well-separated* with three well-separated clusters, the Pareto front obtained from ECM-NSGA-II shows a wide range of resulting clusterings. The most suitable clustering for this dataset is obtained from the leftmost solution on the Pareto front, and we see clusterings at different levels of fuzziness across the Pareto front.

between the clusters in the dataset. As can be seen from the range of clustering solutions identified ranging from the most discrete clusterings on the leftmost side of the Pareto front to the most overlapped clusters on the righthand side, ECM returns a wide range of clustering solutions which can enable us to select the clustering solution with the best suited level of fuzziness for the dataset at hand.

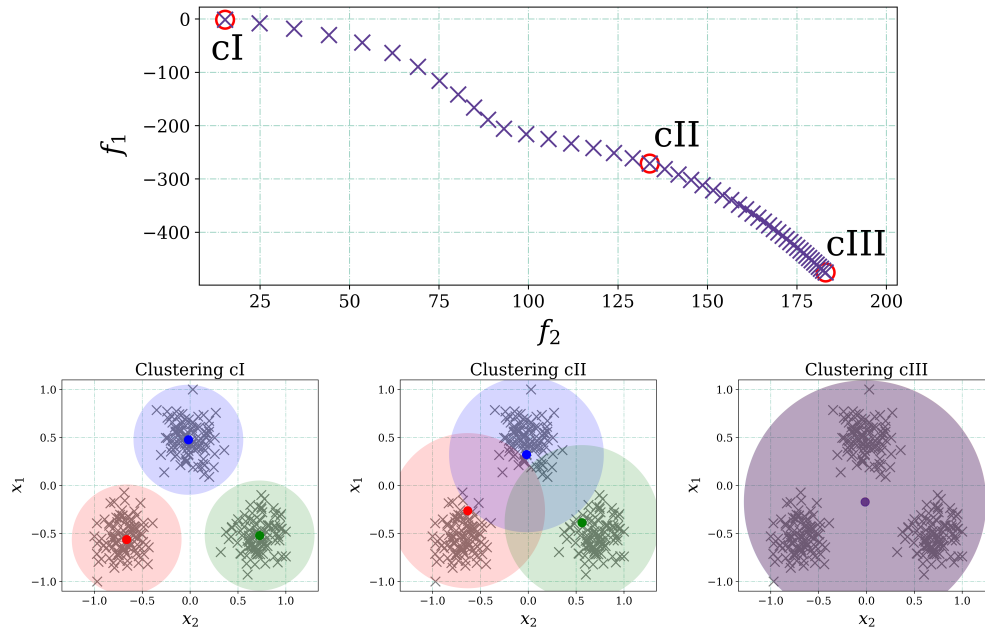


Figure C.15: For the dataset *data1-well-separated* with three well-separated clusters, the ECM-MOEA/D Pareto front shows a wide range of resulting clusterings at different levels of fuzziness. For this dataset, the most appropriate clustering is obtained from the leftmost solution on the Pareto front.

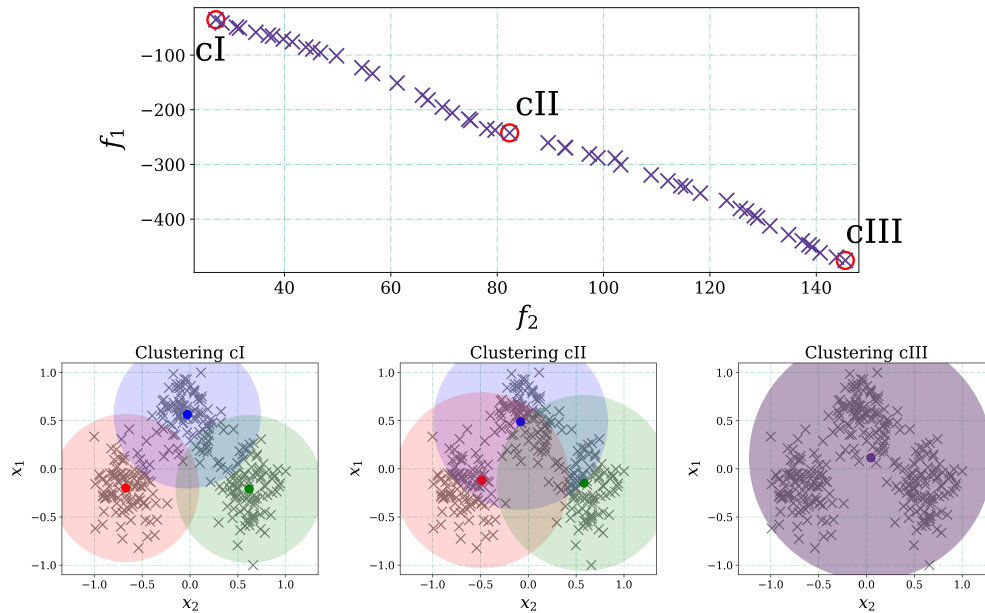


Figure C.16: For the dataset *data2-slight-overlaps* with three slightly overlapped clusters, ECM-NSGA-II results in a wide range of clustering solutions for different levels of fuzziness, three of which are shown above, going from the most discrete clustering obtained in the leftmost solution to the most overlapped clustering solution observed in the rightmost solution. The most suitable clustering can be identified within this range of solutions.

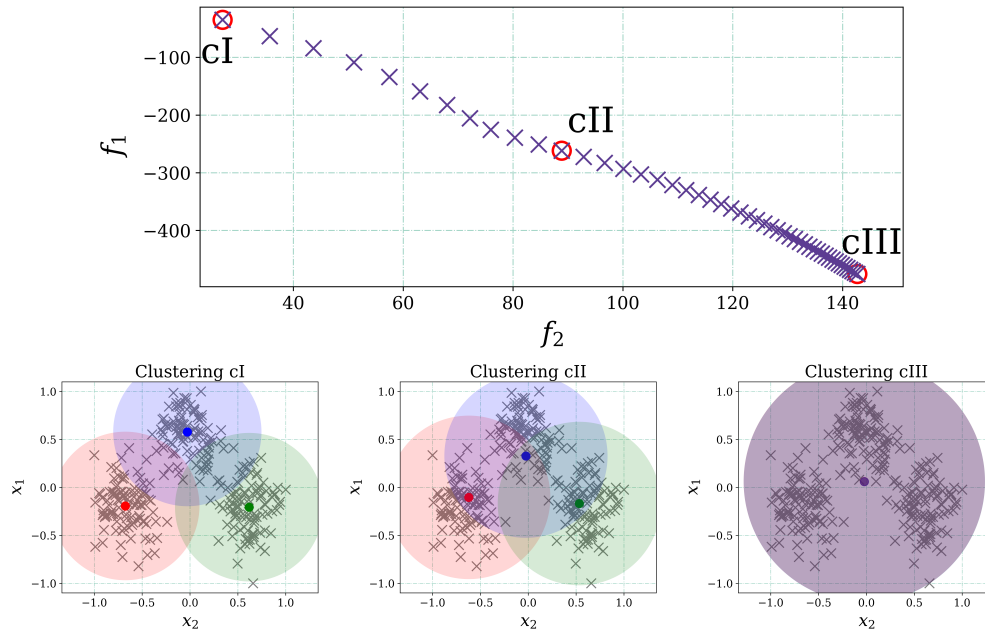


Figure C.17: For ECM-MOEA/D we observe a behaviour similar to the case of ECM-NSGA-II, where for the dataset *data2-slight-overlaps* we obtain a wide range of clustering solutions from the most discrete clusters to the left of the Pareto front, to the most overlapped clusterings obtained in the rightmost solution on the Pareto front.

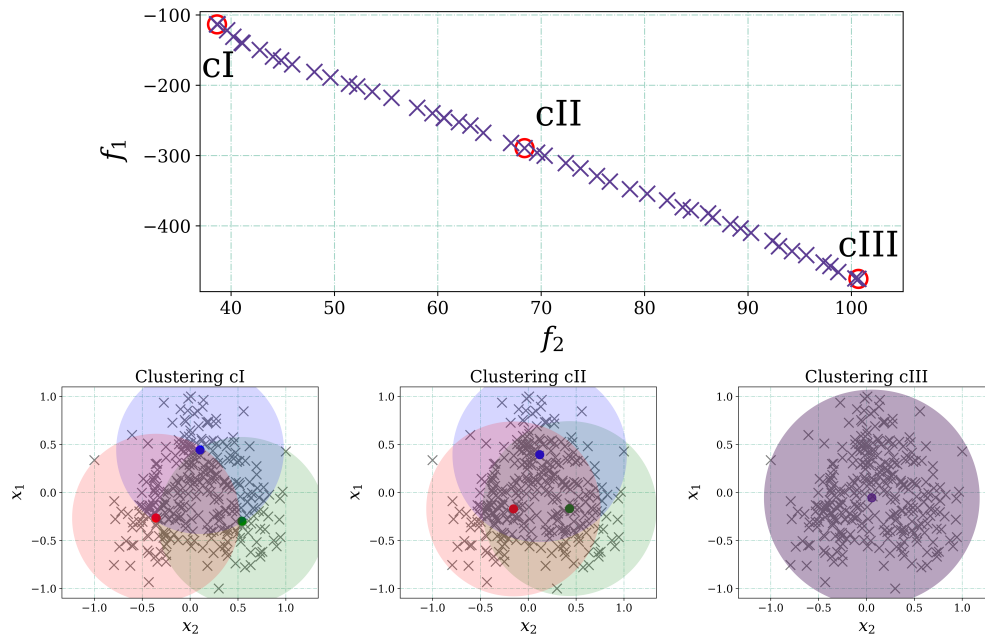


Figure C.18: For the dataset *data3-high-overlaps* with highly overlapped clusters, ECM-NSGA-II results in a Pareto front from which one can obtain a clustering solution with a level of fuzziness that closely matches the underlying degree of overlap of the data.

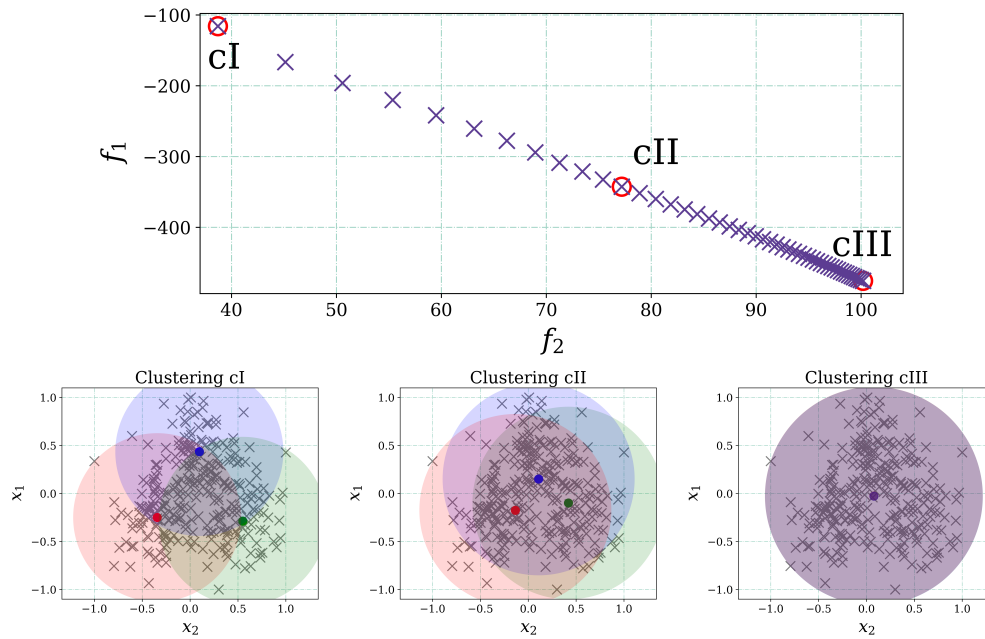


Figure C.19: For the dataset *data3-high-overlaps* with highly overlapped clusters, ECM-MOEA/D results in a wide Pareto front of solutions covering a wide range of levels of fuzziness, from which we can identify the most suitable clustering solution whose level of fuzziness closely matches the underlying degree of overlap of the data.

C.3 Multiple Contradictory Objectives of ECM versus a Combined Single Objective

In this section we compare the proposed multiple contradictory objective clustering setup of ECM with that of a combined single objective clustering problem. For the ease of the discussions, we reproduce here the two objective functions of ECM that were proposed in Chapter 5.

$$\text{minimize } f_1 = \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \|\mathbf{x}_i - \mathbf{v}_j\|^2, \quad (\text{C.1a})$$

$$\text{maximize } f_2 = - \sum_{i=1}^N \sum_{j=1}^c \mu_{ij} \log(\mu_{ij}). \quad (\text{C.1b})$$

The objective function f_1 aims to identify compact clusters by minimizing the sum of the distances of the data instances to each of the clusters, which are weighted by the fuzzy cluster memberships. Minimizing f_1 leads to the identification of discrete clusters. The objective function f_2 on the other hand, is a measure of the entropy of cluster memberships, maximizing which will lead to completely overlapped clusters. Since f_1 aims to identify discrete clusters whereas f_2 aims to identify overlapped clusters, these two objectives are contradictory, which we prove in Chapter 5. The simultaneous optimization of these two objectives by the evolutionary multi-objective optimization algorithms of ECM-NSGA-II and ECM-MOEA/D result in a wide Pareto front of clustering solutions with different levels of fuzziness across the Pareto front. The leftmost solution of the Pareto front yields the clustering with the lowest levels of fuzziness, using which one can identify discrete clusters. The rightmost solution of the Pareto front is a clustering with the highest levels of fuzziness, thus identifying completely overlapped clusters. In between these two solutions, the Pareto front provides a wide range of clustering solutions corresponding to different levels of fuzziness. For a dataset at hand, the most suitable clustering with a level of fuzziness that closely matches the underlying degree of overlap between the clusters in the dataset can thus be identified from this Pareto front.

However, when these two objectives are combined into a single objective, we lose this capability of identifying a wide range of clustering solutions corresponding to different levels of fuzziness. If the two objectives are combined to form a single objective $f = f_1 - f_2$, then both the objectives aim to identify the most discrete clusters. When an evolutionary algorithm is used to optimize this objective, we obtain a population of the same clustering solutions, that identify discrete clusters on the dataset. This is shown in Figure C.20, where compared to the wide Pareto front of ECM containing clusterings at different levels of fuzziness, the combined single objective yields a population of the same solutions identifying only discrete clusters. Similarly, if we consider a combined single objective $f' = f_2 - f_1$, we will obtain only clustering solutions where all the clusters are completely overlapped.

Thus the advantage of using multiple contradictory objectives in ECM is to yield a wide Pareto front of clustering solutions at different levels of fuzziness, from which a clustering can be selected with a suitable level of fuzziness corresponding to the underlying degree of overlap among the clusters in the data at hand.

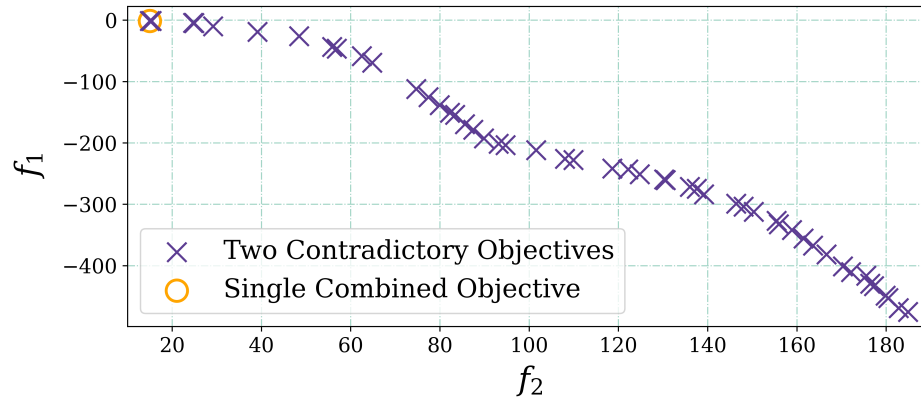


Figure C.20: For the dataset *data1-well-separated* shown in Figure C.13(a), the use of the two contradictory objectives of f_1 and f_2 described in eqns. (C.1a) and (C.1b) enables ECM-NSGA-II to identify a wide Pareto front of clustering solutions at different levels of fuzziness, from which one can identify a clustering that closely matches the underlying degree of overlap of the dataset at hand. In contrast, if the two objectives are combined to form a single objective of $f = f_1 - f_2$ which are not contradicting, an evolutionary optimization algorithm will converge to give all equivalent solutions of a clustering with discrete clusters, as observed on the leftmost side of the Pareto front where ECM also obtains its most discrete clustering solutions.

List of Publications

- A. Gupta and S. Das. On the Unification of k-Harmonic Means and Fuzzy c-Means Clustering Problems under Kernelization. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR 2017)*, pages 1–6, 2017. doi: 10.1109/ICAPR.2017.8593078.
- A. Gupta and S. Das. Transfer Clustering using a Multiple Kernel Metric Learned under Multi-Instance Weak Supervision. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021. doi: 10.1109/TETCI.2021.3110526.
- A. Gupta, S. Datta, and S. Das. Fast automatic estimation of the number of clusters from the minimum inter-center distance for k-means clustering. *Pattern Recognition Letters*, 116: 72–79, 2018. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2018.09.003>.
- A. Gupta, S. Datta, and S. Das. Fuzzy Clustering to Identify Clusters at Different Levels of Fuzziness: An Evolutionary Multiobjective Optimization Approach. *IEEE Transactions on Cybernetics*, 51(5):2601–2611, 2021. doi: 10.1109/TCYB.2019.2907002.
- (*Revised*), A. Gupta, and S. Das. Improved Efficient Model Selection for Sparse Hard and Fuzzy Center-Based Clustering. *Information Sciences*.

References

- M. Abavisani, A. Naghizadeh, D. Metaxas, and V. Patel. Deep Subspace Clustering with Data Augmentation. In *Advances in Neural Information Processing Systems*, volume 33, pages 10360–10370, 2020. [11](#)
- M. Abdolali and M. Rahmati. Neither global nor local: A hierarchical robust subspace clustering for image data. *Information Sciences*, 514:333–353, 2020. [54](#)
- H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. [25](#), [65](#), [117](#), [129](#)
- O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243 – 256, 2013. [19](#), [64](#)
- E. Arias-Castro and X. Pu. A simple approach to sparse clustering. *Computational Statistics & Data Analysis*, 105:217 – 228, 2017. [54](#)
- D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007. [27](#)
- P. Awasthi, M. F. Balcan, and K. Voevodski. Local Algorithms for Interactive Clustering. *Journal of Machine Learning Research*, 18(1):75–109, Jan. 2017. [14](#)
- M. Azizyan, A. Singh, and L. Wasserman. Efficient Sparse Clustering of High-Dimensional Non-spherical Gaussian Mixtures. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 37–45. PMLR, 2015. [9](#)
- L. Bai and J. Liang. Sparse Subspace Clustering with Entropy-Norm. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 561–568. PMLR, 2020. [10](#)
- L. Bai, J. Liang, and F. Cao. Semi-Supervised Clustering with Constraints of Different Types from Multiple Information Sources. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. [14](#), [98](#)
- E. Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013. [13](#), [95](#)

-
- S. Basu, A. Banerjee, and R. J. Mooney. Active Semi-Supervision for Pairwise Constrained Clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 333–344. SIAM, 2004a. [13](#), [95](#)
- S. Basu, M. Bilenko, and R. J. Mooney. A Probabilistic Framework for Semi-Supervised Clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004b. [98](#)
- S. Basu, M. Bilenko, A. Banerjee, R. J. Mooney, and M. Bilenko. *Probabilistic Semi-Supervised Clustering with Constraints*, pages 71–98. MIT Press, September 2006. [13](#), [95](#)
- S. Basu, D. Fisher, S. Drucker, and H. Lu. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1), Jul. 2010. [95](#)
- Y. Bengio. Deep Learning of Representations for Unsupervised and Transfer Learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012. [97](#)
- A. M. Bensaïd, L. O. Hall, J. C. Bezdek, L. P. Clarke, M. L. Silbiger, J. A. Arrington, and R. F. Murtagh. Validity-Guided (Re)Clustering with Applications to Image Segmentation. *IEEE Transactions on Fuzzy Systems*, 4(2):112–123, 1996. [25](#), [65](#), [121](#), [130](#)
- J. C. Bezdek. Cluster Validity with Fuzzy Sets. *Journal of Cybernetics*, 3(3):58–73, 1973. [4](#), [25](#), [72](#)
- J. C. Bezdek. Mathematical Models for Systematics and Taxonomy. In *Eighth International Conference on Numerical Taxonomy*, volume 3, pages 143–166, 1975. [25](#), [65](#), [118](#), [129](#)
- J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981. [40](#), [73](#)
- J. C. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191 – 203, 1984. [4](#), [12](#), [72](#), [73](#)
- J. C. Bezdek, M. Moshtaghi, T. Runkler, and C. Leckie. The Generalized C Index for Internal Fuzzy Cluster Validity. *IEEE Transactions on Fuzzy Systems*, 24(6):1500–1512, 2016. [87](#)
- M. Bilenko, S. Basu, and R. J. Mooney. Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04*, page 11, 2004. [95](#)
- X. bin Zhi, J. lun Fan, and F. Zhao. Fuzzy Linear Discriminant Analysis-guided maximum entropy fuzzy clustering algorithm. *Pattern Recognition*, 46(6):1604 – 1615, 2013. [77](#)
- F. Bourgeois and J.-C. Lassalle. An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. *Communications of the ACM*, 14(12):802–804, 1971. [102](#)

-
- M. Bressan, N. Cesa-Bianchi, A. Paudice, and F. Vitale. Correlation Clustering with Adaptive Similarity Queries. In *Advances in Neural Information Processing Systems*, volume 32, 2019. [14](#)
- Q.-T. Bui, B. Vo, V. Snasel, W. Pedrycz, T.-P. Hong, N.-T. Nguyen, and M.-Y. Chen. SFCM: A Fuzzy Clustering Algorithm of Extracting the Shape Information of Data. *IEEE Transactions on Fuzzy Systems*, 29(1):75–89, 2021. [12](#)
- T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974. [25](#), [65](#), [117](#), [129](#)
- S. Chakraborty and S. Das. Detecting Meaningful Clusters from High-dimensional Data: A Strongly Consistent Sparse Center-based Clustering Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. [9](#)
- S. Chakraborty, D. Paul, S. Das, and J. Xu. Entropy Weighted Power k-Means Clustering. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pages 691–701, 2020. [9](#)
- J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan. Deep Self-Evolution Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):809–823, 2020. [116](#)
- X. Chang, Q. Wang, Y. Liu, and Y. Wang. Sparse Regularization in Fuzzy c -Means for High-Dimensional Data Clustering. *IEEE Transactions on Cybernetics*, 47(9):2616–2627, 2017a. [9](#), [54](#), [56](#)
- Y. Chang, J. Chen, M. H. Cho, P. J. Castaldi, E. K. Silverman, and J. G. Dy. Clustering from Multiple Uncertain Experts. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, volume 54, pages 28–36, 2017b. [14](#)
- Y. Chang, J. Chen, M. H. Cho, P. J. Castaldi, E. K. Silverman, and J. G. Dy. Multiple Clustering Views from Multiple Uncertain Experts. In *Proceedings of the 34th International Conference on Machine Learning*, page 674–683, 2017c. [14](#)
- L. Chen, C. L. P. Chen, and M. Lu. A Multiple-Kernel Fuzzy C-Means Algorithm for Image Segmentation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(5):1263–1274, 2011. [93](#), [99](#)
- M. Chen, L. Huang, C. Wang, and D. Huang. Multi-View Clustering in Latent Embedding Space. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3513–3520, 2020a. [116](#)
- Y. Chen and A. Zhou. Moea/d with an improved multi-dimensional mapping coding scheme for constrained multi-objective portfolio optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1742–1749, 2019. [75](#)
- Y. Chen, C.-G. Li, and C. You. Stochastic Sparse Subspace Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020b. [10](#)

-
- S. K. Choy, S. Y. Lam, K. W. Yu, W. Y. Lee, and K. T. Leung. Fuzzy model-based clustering and its application in image segmentation. *Pattern Recognition*, 68:141 – 157, 2017. 77
- G. Cleuziou and J. G. Moreno. Kernel methods for point symmetry-based clustering. *Pattern Recognition*, 48(9):2812 – 2830, 2015. 40
- R. N. Dave. Validating fuzzy partitions obtained through c-shells clustering. *Pattern Recognition Letters*, 17(6):613–623, 1996. 25, 65, 119, 121, 130
- D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979. 25, 65, 118, 129
- R. C. de Amorim. A survey on Feature Weighting based K-Means Algorithms. *Journal of Classification*, 33(2):210–242, Jul 2016. 54
- F. de A.T. de Carvalho, E. C. Simões, L. V. Santana, and M. R. Ferreira. Gaussian kernel c-means hard clustering algorithms with automated computation of the width hyperparameters. *Pattern Recognition*, 79:370–386, 2018. 7
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. 75, 78, 79
- D. Dembélé and P. Kastner. Fuzzy C-means method for clustering microarray data. *Bioinformatics*, 19(8):973–980, 2003. 73
- T. Deng, D. Ye, R. Ma, H. Fujita, and L. Xiong. Low-rank local tangent space embedding for subspace clustering. *Information Sciences*, 508:1–21, 2020. 54
- Z. Deng, K.-S. Choi, Y. Jiang, J. Wang, and S. Wang. A survey on soft subspace clustering. *Information Sciences*, 348:84–106, 2016. 54
- Z. Deng, Y. Jiang, F. Chung, H. Ishibuchi, K. Choi, and S. Wang. Transfer Prototype-Based Fuzzy Clustering. *IEEE Transactions on Fuzzy Systems*, 24(5):1210–1232, 2016. 13, 95
- S. Dey, S. Das, and R. Mallipeddi. The Sparse MinMax k-Means Algorithm for High-Dimensional Clustering. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2103–2110, 2020. 9
- D. Dheeru and E. Karra Taniskidou. UCI Machine Learning Repository, 2017. URL <http://archive.ics.uci.edu/ml>. 37, 47, 68, 83
- I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: Spectral Clustering and Normalized Cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 551–556, 2004. 40, 99
- T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997. 95
- E. Dimitriadou, S. Dolničar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–159, Mar 2002. 19, 24

-
- C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma. Subspace clustering of high dimensional data. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 517–521, 2004. [54](#)
- C. Döring, M.-J. Lesot, and R. Kruse. Data analysis with fuzzy clustering methods. *Computational Statistics & Data Analysis*, 51(1):192 – 214, 2006. [73](#)
- L. Du, P. Zhou, L. Shi, H. Wang, M. Fan, W. Wang, and Y.-D. Shen. Robust Multiple Kernel K-means Using $l_{2,1}$ -Norm. In *Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015. [95](#), [107](#), [115](#)
- M. Du, S. Ding, and Y. Xue. A novel density peaks clustering algorithm for mixed data. *Pattern Recognition Letters*, 97:46 – 53, 2017. [19](#)
- J. C. Dunn. A Fuzzy Relative of the ISODATA process and its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1973. [4](#), [12](#), [25](#), [40](#), [72](#), [73](#), [119](#)
- T. Durand, T. Mordan, N. Thome, and M. Cord. WILDCAT: Weakly Supervised Learning of Deep Convnets for Image Classification, Pointwise Localization and Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 642–651, 2017. [95](#)
- K. Efimov, L. Adamyan, and V. Spokoiny. Adaptive Nonparametric Clustering. *IEEE Transactions on Information Theory*, 65(8):4875–4892, 2019. [6](#), [115](#)
- E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, Theory, and Applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013. [9](#), [54](#), [95](#)
- Q. Feng, L. Chen, C. L. P. Chen, and L. Guo. Deep Fuzzy Clustering — A Representation Learning Approach. *IEEE Transactions on Fuzzy Systems*, 28(7):1420–1433, 2020. [12](#)
- M. R. Ferreira and F. de A.T. de Carvalho. Kernel fuzzy c-means with automatic variable weighting. *Fuzzy Sets and Systems*, 237:1–46, 2014. [40](#), [100](#)
- M. R. Ferreira, F. de A.T. de Carvalho, and E. C. Simões. Kernel-based hard clustering methods with kernelization of the metric and automatic weighting of the variables. *Pattern Recognition*, 51:310 – 321, 2016. [40](#)
- N. Flammarion, B. Paliappan, and F. Bach. Robust Discriminative Clustering with Sparse Regularizers. *Journal of Machine Learning Research*, 18(80):1–50, 2017. [9](#)
- B. J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976, 2007. [19](#)
- A. Fujita, D. Y. Takahashi, and A. G. Patriota. A non-parametric method to estimate the number of clusters. *Computational Statistics & Data Analysis*, 73:27–39, 2014. [21](#), [25](#), [123](#)
- Y. Fukuyama and M. Sugeno. A new method of choosing the number of clusters for the fuzzy c-means method. *Proceedings of the Fifth Fuzzy Systems Symposium*, pages 247–250, 1989. [25](#), [65](#), [119](#), [129](#)

-
- M. E. Futschik and B. Carlisle. Noise-robust soft clustering of gene expression time-course data. *Journal of Bioinformatics and Computational Biology*, 03(04):965–988, 2005. [73](#)
- S. Gaynor and E. Bair. Identification of relevant subtypes via preweighted sparse clustering. *Computational Statistics & Data Analysis*, 116:139 – 154, 2017. [54](#)
- M. G. Genton. Classes of Kernels for Machine Learning: A Statistics Perspective. *Journal of Machine Learning Research*, 2:299–312, 2002. [101](#)
- M. Girolami. Mercer Kernel-Based Clustering in Feature Space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002. [99](#)
- W. Gong, R. Zhao, and S. Grünewald. Structured sparse k-means clustering via Laplacian smoothing. *Pattern Recognition Letters*, 112:63 – 69, 2018. [67](#), [137](#)
- M. B. Gorzałczany and F. Rudziński. Generalized Self-Organizing Maps for Automatic Determination of the Number of Clusters and Their Multiprototypes in Cluster Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):2833–2845, 2018. [6](#)
- D. Graves and W. Pedrycz. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. *Fuzzy Sets and Systems*, 161(4):522 – 543, 2010. [40](#)
- N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5):1834–1844, 2008. [13](#), [95](#)
- L. Gröll and J. Jäkel. A New Convergence Proof of Fuzzy c-Means. *IEEE Transactions on Fuzzy Systems*, 13(5):717–720, 2005. [46](#)
- A. Gupta and S. Das. On the unification of k-harmonic means and fuzzy c-means clustering problems under kernelization. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR 2017)*, pages 1–6, 2017. [100](#)
- M. Halkidi and M. Vazirgiannis. Clustering Validity Assessment: Finding the optimal partitioning of a data set. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001)*, pages 187–194, 2001. [25](#), [120](#)
- G. Hamerly and C. Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM '02*, pages 600–607, 2002. [40](#)
- G. Hamerly and C. Elkan. Learning the k in k-means. In *Proceedings of the 16th International Conference on Neural Information Processing Systems, NIPS'03*, pages 281–288, 2003. [19](#)
- K. Han, A. Vedaldi, and A. Zisserman. Learning to Discover Novel Visual Categories via Deep Transfer Clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. [13](#), [95](#)
- Y. Han, K. Yang, Y. Yang, and Y. Ma. Localized Multiple Kernel Learning With Dynamical Clustering and Matrix Regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):486–499, 2018a. [8](#), [95](#)

-
- Y. Han, Y. Yang, X. Li, Q. Liu, and Y. Ma. Matrix-Regularized Multiple Kernel Learning via (r, p) Norms. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4997–5007, 2018b. [95](#)
- E. Hancer and D. Karaboga. A Comprehensive Survey of Traditional, Merge-Split and Evolutionary Approaches Proposed for Determination of Cluster Number. *Swarm and Evolutionary Computation*, 32:49 – 67, 2017. [19](#)
- J. Handl and J. Knowles. An Evolutionary Approach to Multiobjective Clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76, 2007. [19](#), [64](#), [81](#), [139](#)
- J. A. Hartigan. Statistical theory in clustering. *Journal of classification*, 2(1):63–76, 1985. [25](#), [120](#)
- T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008. [101](#)
- X. Hu, Y. Shen, W. Pedrycz, X. Wang, A. Gacek, and B. Liu. Identification of fuzzy rule-based models with collaborative fuzzy clustering. *IEEE Transactions on Cybernetics*, pages 1–14, 2021. [12](#)
- H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen. Multiple Kernel Fuzzy Clustering. *IEEE Transactions on Fuzzy Systems*, 20(1):120–134, 2012a. [99](#)
- J. Z. Huang, M. K. Ng, Hongqiang Rong, and Zichen Li. Automated Variable Weighting in k-Means Type Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005. [53](#), [95](#)
- M. Huang, Z. Xia, H. Wang, Q. Zeng, and Q. Wang. The range of the value for the fuzzifier of the fuzzy c-means algorithm. *Pattern Recognition Letters*, 33(16):2280 – 2284, 2012b. [74](#)
- L. Hubert and P. Arabie. Comparing Partitions. *Journal of Classification*, 2(1):193–218, 1985. [46](#), [82](#), [107](#)
- A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. [3](#), [19](#), [39](#)
- A. Jalali and R. Willett. Subspace Clustering via Tangent Cones. In *Advances in Neural Information Processing Systems*, volume 30, 2017. [10](#)
- P. Jha, A. Tiwari, N. Bharill, M. Ratnaparkhe, M. Mounika, and N. Nagendra. A Novel Scalable Kernelized Fuzzy Clustering Algorithms Based on In-Memory Computation for Handling Big Data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–12, 2020. [99](#)
- P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid. Deep Subspace Clustering Networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. [11](#)
- X. Ji, J. F. Henriques, and A. Vedaldi. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019. [95](#)

-
- H. Jia and Y. Cheung. Subspace Clustering of Categorical and Numerical Data With an Unknown Number of Clusters. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3308–3325, 2018. [10](#), [54](#)
- W. Jiang and F.-I. Chung. Transfer Spectral Clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 789–803, 2012. [13](#), [95](#)
- R. Jin, S. Wang, and Z.-H. Zhou. Learning a Distance Metric from Multi-instance Multi-label Data. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 896–902, 2009. [95](#)
- X. Jing, Z. Yan, Y. Shen, W. Pedrycz, and J. Yang. A group-based distance learning method for semisupervised fuzzy clustering. *IEEE Transactions on Cybernetics*, pages 1–14, 2020. [14](#)
- A. Jitta and A. Klami. On Controlling the Size of Clusters in Probabilistic Clustering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3350–3357, 2018. [116](#)
- Z. Kang, C. Peng, and Q. Cheng. Twin Learning for Similarity and Clustering: A Unified Kernel Approach. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI’17, page 2080–2086, 2017. [8](#)
- Z. Kang, X. Lu, J. Yi, and Z. Xu. Self-weighted Multiple Kernel Learning for Graph-based Clustering and Semi-supervised Classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2312–2318, 2018. [8](#)
- Z. Kang, W. Zhou, Z. Zhao, J. Shao, M. Han, and Z. Xu. Large-Scale Multi-View Subspace Clustering in Linear Time. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4412–4419, 2020. [10](#)
- N. B. Karayiannis. MECA: maximum entropy clustering algorithm. In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, pages 630–635, 1994. [77](#)
- D.-W. Kim, K. Lee, D. Lee, and K. H. Lee. A kernel-based subtractive clustering method. *Pattern Recognition Letters*, 26(7):879–891, 2005. [40](#)
- F. Klawonn and F. Höppner. *What Is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier*. Springer, Berlin, Heidelberg, 2003. [73](#)
- F. Klawonn and F. Höppner. An alternative approach to the fuzzifier in fuzzy clustering to obtain better clustering. In *Proceedings of the 3rd Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2003)*, pages 730–734, Sept 2003. [74](#)
- Y. Kondo, M. Salibian-Barrera, and R. Zamar. A robust and sparse K-means clustering algorithm. *arXiv e-prints*, art. arXiv:1201.6082, Jan 2012. [67](#), [137](#)
- R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993. [12](#)

-
- H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. [102](#)
- B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009. [13](#), [95](#)
- S. Kushagra, S. Ben-David, and I. F. Ilyas. Semi-supervised clustering for de-duplication. In *The 22nd International Conference on Artificial Intelligence and Statistics*, volume 89, pages 1659–1667, 2019. [14](#)
- Y. Lai, S. He, Z. Lin, F. Yang, Q. Zhou, and X. Zhou. An Adaptive Robust Semi-Supervised Clustering Framework Using Weighted Consensus of Random k -Means Ensemble. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1877–1890, 2021. [14](#)
- M. T. Law, Y. Yu, R. Urtasun, R. S. Zemel, and E. P. Xing. Efficient Multiple Instance Metric Learning using Weakly Supervised Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 576–584, 2017. [95](#), [98](#), [99](#), [100](#), [102](#), [105](#)
- J. Li, Y. Kong, and Y. Fu. Sparse Subspace Clustering by Learning Approximation 0 Codes. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2189–2195, 2017. [10](#)
- M. Li, X. Liu, L. Wang, Y. Dou, J. Yin, and E. Zhu. Multiple Kernel Clustering with Local Kernel Alignment Maximization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1704–1710, 2016. [8](#), [95](#), [99](#)
- Q. Li, N. Mitianoudis, and T. Stathaki. Spatial kernel K-harmonic means clustering for multi-spectral image segmentation. *IET Image Processing*, 1(2):156–167, 2007. [40](#)
- R. Li, C. Zhang, Q. Hu, P. Zhu, and Z. Wang. Flexible Multi-View Representation Learning for Subspace Clustering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2916–2922, 2019. [10](#)
- R.-P. Li and M. Mukaidono. A maximum-entropy approach to fuzzy clustering. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems.*, volume 4, pages 2227–2232, 1995. [74](#), [76](#), [82](#)
- W. Li, J. Hannig, and S. Mukherjee. Subspace Clustering through Sub-Clusters. *Journal of Machine Learning Research*, 22(53):1–37, 2021. [10](#)
- Y. Li, Y. Wang, D.-J. Yu, N. Ye, P. Hu, and R. Zhao. ASCENT: Active Supervision for Semi-Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):868–882, 2020. [14](#)
- J. Liang, J. Yang, M.-M. Cheng, P. L. Rosin, and L. Wang. Simultaneous Subspace Clustering and Cluster Number Estimating Based on Triplet Relationship. *IEEE Transactions on Image Processing*, 28(8):3973–3985, 2019. [10](#)
- Z. Liang and P. Chen. Delta-density based clustering with a divide-and-conquer strategy: 3DC Clustering. *Pattern Recognition Letters*, 73:52 – 59, 2016. [19](#)

-
- W. Lin, Z. He, and M. Xiao. Balanced Clustering: A Uniform Model and Fast Algorithm. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pages 2987–2993, 2019. [116](#)
- H. Liu, Z. Tao, and Y. Fu. Partition Level Constrained Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2469–2483, 2018. [14](#)
- J. Liu, X. Liu, J. Xiong, Q. Liao, S. Zhou, S. Wang, and Y. Yang. Optimal Neighborhood Multiple Kernel Clustering with Adaptive Local Kernels. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2020a. [8](#), [95](#), [99](#), [105](#)
- W. Liu, X. Shen, and I. Tsang. Sparse Embedded k-Means Clustering. In *Advances in Neural Information Processing Systems*, volume 30, 2017a. [11](#)
- X. Liu, Y. Dou, J. Yin, L. Wang, and E. Zhu. Multiple Kernel k -Means Clustering with Matrix-Induced Regularization. In *Proceedings of the thirtieth AAAI conference on artificial intelligence*, pages 1888–1894, 2016. [8](#), [95](#), [99](#)
- X. Liu, S. Zhou, Y. Wang, M. Li, Y. Dou, E. Zhu, and J. Yin. Optimal Neighborhood Kernel Clustering with Multiple Kernels. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, page 2266–2272, 2017b. [8](#), [49](#), [93](#), [95](#), [99](#), [105](#)
- X. Liu, X. Zhu, M. Li, C. Tang, E. Zhu, J. Yin, and W. Gao. Efficient and Effective Incomplete Multi-View Clustering. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI’19*, pages 4392–4399, 2019. [8](#), [95](#), [99](#)
- X. Liu, X. Zhu, M. Li, L. Wang, C. Tang, J. Yin, D. Shen, H. Wang, and W. Gao. Late Fusion Incomplete Multi-View Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2410–2423, 2019. [8](#), [95](#), [99](#)
- X. Liu, M. Li, C. Tang, J. Xia, J. Xiong, L. Liu, M. Kloft, and E. Zhu. Efficient and Effective Regularized Incomplete Multi-View Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2020b. [8](#), [95](#), [99](#)
- X. Liu, L. Wang, X. Zhu, M. Li, E. Zhu, T. Liu, L. Liu, Y. Dou, and J. Yin. Absent Multiple Kernel Learning Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1303–1316, 2020c. [95](#)
- X. Liu, L. Wang, X. Zhu, M. Li, E. Zhu, T. Liu, L. Liu, Y. Dou, and J. Yin. Absent Multiple Kernel Learning Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1303–1316, 2020a. [8](#)
- X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao. Multiple Kernel k -Means with Incomplete Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(5):1191–1204, 2020b. [8](#), [105](#)
- G. Lizárraga, A. Hernández, and S. Botello. *A Set of Test Cases for Performance Measures in Multiobjective Optimization*, pages 429–439. Springer Berlin Heidelberg, 2008. [83](#), [85](#)
- S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. [3](#), [40](#)

-
- S. Louhichi, M. Gzara, and H. Ben-Abdallah. Unsupervised varied density based clustering algorithm using spline. *Pattern Recognition Letters*, 93:48–57, 2017. [19](#)
- C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan. Subspace clustering by block diagonal representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):487–501, 2019. [10](#)
- Y. Lu, L. Wang, J. Lu, J. Yang, and C. Shen. Multiple kernel clustering based on centered kernel alignment. *Pattern Recognition*, 47(11):3656 – 3664, 2014. [95](#)
- J. M. Luna-Romera, M. Martínez-Ballesteros, J. García-Gutiérrez, and J. C. Riquelme. External clustering validity index based on chi-squared statistical test. *Information Sciences*, 487:1 – 17, 2019. [64](#)
- S. Luo, C. Zhang, W. Zhang, and X. Cao. Consistent and Specific Multi-View Subspace Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. [10](#)
- J. Lv, Z. Kang, X. Lu, and Z. Xu. Pseudo-Supervised Deep Subspace Clustering. *IEEE Transactions on Image Processing*, 30:5252–5263, 2021. [116](#)
- D. MacDonald and C. Fyfe. The Kernel Self Organising Map. In *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, volume 1, pages 317–320, 2000. [40](#)
- J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967. [3](#), [19](#), [39](#)
- D. Marin, M. Tang, I. B. Ayed, and Y. Boykov. Kernel Clustering: Density Biases and Solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):136–147, 2019. [7](#)
- S. Matsushima and M. Brbic. Selective Sampling-based Scalable Sparse Subspace Clustering. In *Advances in Neural Information Processing Systems*, volume 32, 2019. [10](#)
- U. Maulik and S. Bandyopadhyay. Performance Evaluation of Some Clustering Algorithms and Validity Indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002. [25](#), [65](#), [120](#), [130](#)
- G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, Jun 1985. [19](#), [24](#)
- M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost. Explainable k-Means and k-Medians Clustering. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pages 7055–7065, 2020. [115](#)
- S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):4610–4617, 2019. [11](#)

-
- A. Mukhopadhyay and U. Maulik. Unsupervised Pixel Classification in Satellite Imagery Using Multiobjective Fuzzy Clustering Combined With SVM Classifier. *IEEE Transactions on Geoscience and Remote Sensing*, 47(4):1132–1138, 2009. [80](#)
- A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. Clustering using Multi-objective Genetic Algorithm and its Application to Image Segmentation. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2678–2683, 2006. [80](#)
- A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello. Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part II. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35, 2014. [74](#)
- K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001. [40](#)
- K. Nguyen, N. Dam, T. Le, T. D. Nguyen, and D. Q. Phung. Clustering Induced Kernel Learning. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML*, volume 95, pages 129–144, 2018. [8](#)
- F. Nie, G. Cai, and X. Li. Multi-View Clustering and Semi-Supervised Classification with Adaptive Neighbours. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017. [14](#)
- F. Nie, H. Zhang, R. Wang, and X. Li. Semi-supervised Clustering via Pairwise Constrained Optimal Graph. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3160–3166, 2020. [14](#)
- I. Ozkan and I. Turksen. Upper and lower values for the level of fuzziness in FCM. *Information Sciences*, 177(23):5143 – 5152, 2007. [73](#)
- M. K. Pakhira, S. Bandyopadhyay, and U. Maulik. Validity index for crisp and fuzzy clusters. *Pattern recognition*, 37(3):487–501, 2004. [25](#), [65](#), [121](#), [130](#)
- N. R. Pal and J. C. Bezdek. On Cluster Validity for the Fuzzy c-Means Model. *IEEE Transactions on Fuzzy Systems*, 3(3):370–379, 1995. [73](#)
- N. R. Pal and K. Sarkar. What and When Can We Gain From the Kernel Versions of C-Means Algorithm? *IEEE Transactions on Fuzzy Systems*, 22(2):363–379, 2014. [52](#)
- S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. [95](#), [97](#)
- S. J. Pan, J. T. Kwok, and Q. Yang. Transfer Learning via Dimensionality Reduction. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI’08*, page 677–682, 2008. [97](#)
- D. Pelleg and A. W. Moore. X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning, ICML ’00*, page 727–734, 2000. [60](#)

-
- X. Peng, J. Feng, J. Lu, W.-Y. Yau, and Z. Yi. Cascade Subspace Clustering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2478–2484, 2017. [11](#)
- X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan. Deep Subspace Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5509–5521, 2020. [11](#)
- G. Peters, F. Crespo, P. Lingras, and R. Weber. Soft clustering - Fuzzy and rough approaches and their extensions and derivatives. *International Journal of Approximate Reasoning*, 54(2):307 – 322, 2013. [73](#)
- V. Petrosyan and A. Proutiere. Viral Clustering: A Robust Method to Extract Structures in Heterogeneous Datasets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016. [6](#)
- X. Qiu, Y. Qiu, G. Feng, and P. Li. A sparse fuzzy c-means algorithm based on sparse clustering framework. *Neurocomputing*, 157:290–295, 2015. [54](#), [56](#)
- B. Qu, J. Liang, Y. Zhu, Z. Wang, and P. Suganthan. Economic emission dispatch problems with stochastic wind power using summation based multi-objective evolutionary algorithm. *Information Sciences*, 351(Supplement C):48–66, 2016. [87](#)
- M. Rahmani and G. Atia. Innovation Pursuit: A New Approach to the Subspace Clustering Problem. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2874–2882, 2017. [10](#)
- P. Rathore, Z. Ghafoori, J. C. Bezdek, M. Palaniswami, and C. Leckie. Approximating Dunn’s Cluster Validity Indices for Partitions of Big Data. *IEEE Transactions on Cybernetics*, 49(5):1629–1641, 2019. [6](#)
- A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Rè. Training Complex Models with Multi-Task Weak Supervision. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, pages 4763–4771, 2019. [12](#), [95](#)
- A. Ratner, P. Varma, B. Hancock, and C. Ré. *Weak Supervision: A New Programming Paradigm for Machine Learning*, 2019 (accessed July 16, 2021). URL <https://ai.stanford.edu/blog/weak-supervision/>. [12](#), [95](#)
- A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Rè. Snorkel: Rapid Training Data Creation with Weak Supervision. *The VLDB Journal*, 29:709–730, 2020. [12](#), [95](#)
- M. Ren, P. Liu, Z. Wang, and J. Yi. A Self-Adaptive Fuzzy c-Means Algorithm for Determining the Optimal Number of Clusters. *Computational Intelligence and Neuroscience*, pages 3–15, 2016. [25](#), [65](#), [122](#), [131](#)
- Z. Ren and Q. Sun. Simultaneous Global and Local Graph Structure Preserving for Multiple Kernel Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2020. [8](#), [95](#)
- B. Rezaee. A cluster validity index for fuzzy clustering. *Fuzzy Sets and Systems*, 161(23):3014–3025, 2010. [25](#), [65](#), [124](#), [131](#)

-
- M. R. Rezaee, B. P. Lelieveldt, and J. H. Reiber. A new cluster validity index for the fuzzy c -mean. *Pattern Recognition Letters*, 19(3):237–246, 1998. [21](#), [25](#), [118](#)
- M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To Transfer or Not To Transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005. [97](#)
- P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. [25](#), [65](#), [123](#), [131](#)
- E. H. Ruspini. A new approach to clustering. *Information and control*, 15(1):22–32, 1969. [12](#)
- E. H. Ruspini, J. C. Bezdek, and J. M. Keller. Fuzzy Clustering: A Historical Perspective. *IEEE Computational Intelligence Magazine*, 14(1):45–55, 2019. [12](#)
- A. Saha and S. Das. Axiomatic generalization of the membership degree weighting function for fuzzy C means clustering: Theoretical development and convergence analysis. *Information Sciences*, 408:129 – 145, 2017. [74](#)
- A. Saha and S. Das. On the unification of possibilistic fuzzy clustering: Axiomatic development and convergence analysis. *Fuzzy Sets and Systems*, 340:73 – 90, 2018. [93](#)
- J. Saha and J. Mukherjee. Cnak: Cluster number assisted k -means. *Pattern Recognition*, 110:107625, 2021. [6](#)
- N. Saini and S. Saha. Multi-objective optimization techniques: a survey of the state-of-the-art and applications. *The European Physical Journal Special Topics*, 230:2319—2335, 2021. [78](#)
- A. Salah, N. Rogovschi, and M. Nadif. Model-based Co-clustering for High Dimensional Sparse Data. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51, pages 866–874, 2016. [115](#)
- K. Sarkar and N. R. Pal. Is It Rational To Partition A Data Set Using Kernel-Clustering? In *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 2600–2605, 2011. [52](#)
- S. Sarkar and A. K. Ghosh. On Perfect Clustering of High Dimension, Low Sample Size Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2257–2272, 2020. [7](#)
- S. M. Savaresi and D. L. Boley. On the Performance of Bisecting K -Means and PDDP. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–14, 2001. [19](#)
- B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998. [42](#), [99](#)
- V. Schwämmle and O. N. Jensen. A simple and fast method to determine the parameters for fuzzy c -means cluster analysis. *Bioinformatics*, 26(22):2841–2848, 2010. [73](#)

-
- G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978. [25](#), [54](#), [117](#)
- X. Shen, W. Liu, I. Tsang, F. Shen, and Q.-S. Sun. Compressed k-means for large-scale clustering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI’17, page 2527–2533, 2017. [11](#)
- Y. Shi, Z. Yu, W. Cao, C. L. P. Chen, H.-S. Wong, and G. Han. Fast and Effective Active Clustering Ensemble Based on Density Peak. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. [14](#)
- R. G. F. Soares, H. Chen, and X. Yao. A Cluster-Based Semisupervised Ensemble for Multi-class Classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(6):408–420, 2017. [13](#), [95](#)
- M. Soltanolkotabi, E. Elhamifar, E. J. Candes, et al. Robust subspace clustering. *The Annals of Statistics*, 42(2):669–699, 2014. [95](#)
- A. Srivastava, M. Baranwal, and S. Salapaka. On the Persistence of Clustering Solutions and True Number of Clusters in a Dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5000–5007, 2019. [6](#)
- C. A. Sugar and G. M. James. Finding the Number of Clusters in a Dataset: An Information-Theoretic Approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003. [25](#), [121](#)
- X. L. Sui, L. Xu, X. Qian, and T. Liu. Convex clustering with metric learning. *Pattern Recognition*, 81:575–584, 2018. [7](#)
- A. Suleman. A convex semi-nonnegative matrix factorisation approach to fuzzy c-means clustering. *Fuzzy Sets and Systems*, 270:90–110, 2015. [74](#)
- D. Sun, K. Toh, and Y. Yuan. Convex Clustering: Model, Theoretical Guarantee and Efficient Algorithm. *Journal of Machine Learning Research*, 22:9:1–9:32, 2021. [116](#)
- X. Sun, M. Cheng, C. Min, and L. Jing. Self-Supervised Deep Multi-View Subspace Clustering. In *Proceedings of The 11th Asian Conference on Machine Learning, ACML*, pages 1001–1016, 2019. [11](#)
- P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. Yuille. PCL: Proposal Cluster Learning for Weakly Supervised Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):176–191, 2020. [14](#)
- Z. Tao, H. Liu, H. Fu, and Y. Fu. Image Cosegmentation via Saliency-Guided Constrained Clustering with Cosine Similarity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI’17, page 4285–4291, 2017. [13](#)
- Z. Tao, H. Liu, H. Fu, and Y. Fu. Multi-View Saliency-Guided Clustering for Image Cosegmentation. *IEEE Transactions on Image Processing*, 28(9):4634–4645, 2019. [13](#)

-
- R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. [54](#)
- R. Tibshirani and G. Walther. Cluster Validation by Prediction Strength. *Journal of Computational and Graphical Statistics*, 14(3):511–528, 2005. [19](#), [25](#), [122](#)
- R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001. [19](#), [24](#), [25](#), [54](#), [65](#), [119](#)
- Q. Tong, X. Li, and B. Yuan. A highly scalable clustering scheme using boundary information. *Pattern Recognition Letters*, 89:1 – 7, 2017. [19](#)
- C. Tsai, H. Lin, J. Taur, and C. Tao. Iris Recognition Using Possibilistic Fuzzy Matching on Local Features. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):150–162, 2012. [93](#)
- M. Tsakiris and R. Vidal. Theoretical Analysis of Sparse Subspace Clustering with Missing Entries. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4975–4984, 2018. [10](#)
- L. C. Vankadara and D. Ghoshdastidar. On the optimality of kernels for high-dimensional clustering. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 2185–2195, 2020. [7](#)
- V. N. Vapnik. *Statistical Learning Theory*. Wiley New York, 1998. [40](#)
- R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011. [9](#), [54](#)
- A. Vouros and E. Vasilaki. A semi-supervised sparse K-Means algorithm. *Pattern Recognition Letters*, 142:65–71, 2021. [9](#)
- J. Wang, A. Suzuki, L. Xu, F. Tian, L. Yang, and K. Yamanishi. Orderly Subspace Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5264–5272, 2019a. [14](#)
- J. Wang, L. Xu, F. Tian, A. Suzuki, C. Zhang, and K. Yamanishi. Attributed Subspace Clustering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3719–3725, 2019b. [10](#)
- S. Wang, A. Gittens, and M. W. Mahoney. Scalable Kernel K-Means Clustering with Nystrom Approximation: Relative-Error Bounds. *Journal of Machine Learning Research*, 20(12):1–49, 2019c. [10](#)
- S. Wang, Z. Chen, S. Du, and Z. Lin. Learning Deep Sparse Regularizers with Applications to Multi-View Clustering and Semi-Supervised Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. [116](#)
- X. Wang, Y. Wang, and L. Wang. Improving fuzzy c-means clustering based on feature-weight learning. *Pattern Recognition Letters*, 25(10):1123–1132, 2004. [54](#)

-
- X. Wang, B. Qian, J. Ye, and I. Davidson. Multi-Objective Multi-View Spectral Clustering via Pareto Optimization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 234–242, 2013. [74](#)
- X. Wang, R. Chen, Z. Zeng, C. Hong, and F. Yan. Robust Dimension Reduction for Clustering with Local Adaptive Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):657–669, 2019. [54](#)
- X. Wang, J. Wang, C. Domeniconi, G. Yu, G. Xiao, and M. Guo. Multiple Independent Subspace Clusterings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5353–5360, 2019. [10](#)
- Y. Wang, X. Liu, Y. Dou, and R. Li. Multiple Kernel Clustering Framework with Improved Kernels. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI’17*, pages 2999–3005, 2017. [8](#), [95](#), [99](#), [105](#)
- S. Wei, J. Wang, G. Yu, C. Domeniconi, and X. Zhang. Multi-View Multiple Clusterings Using Deep Matrix Factorization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pages 6348–6355, 2020. [116](#)
- D. M. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010. [8](#), [54](#), [55](#), [57](#)
- C. Wu and X. Zhang. A Novel Kernelized Total Bregman Divergence-Driven Possibilistic Fuzzy Clustering with Multiple Information Constraints for Image Segmentation. *IEEE Transactions on Fuzzy Systems*, 2021. [12](#)
- C. H. Wu, C. S. Ouyang, L. W. Chen, and L. W. Lu. A New Fuzzy Clustering Validity Index With a Median Factor for Centroid-Based Clustering. *IEEE Transactions on Fuzzy Systems*, 23(3):701–718, 2015. [87](#)
- K.-L. Wu. Analysis of parameter selections for fuzzy c-means. *Pattern Recognition*, 45(1):407–415, 2012. [74](#)
- K.-L. Wu and M.-S. Yang. A cluster validity index for fuzzy clustering. *Pattern Recognition Letters*, 26(9):1275–1291, 2005. [25](#), [65](#), [122](#), [131](#)
- H. Xia, J. Zhuang, and D. Yu. Novel soft subspace clustering with multi-objective evolutionary approach for high-dimensional data. *Pattern Recognition*, 46(9):2562 – 2575, 2013. [74](#)
- X. L. Xie and G. Beni. A Validity Measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991. [20](#), [25](#), [65](#), [80](#), [123](#), [132](#)
- E. Xing, M. Jordan, S. J. Russell, and A. Ng. Distance Metric Learning with Application to Clustering with Side-Information. In *Advances in Neural Information Processing Systems*, volume 15 of *NIPS’02*. MIT Press, 2003. [95](#)
- C. Xiong, D. M. Johnson, and J. J. Corso. Active Clustering with Model-Based Uncertainty Reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):5–17, 2017. [14](#), [95](#)

-
- S. Xiong, J. Azimi, and X. Z. Fern. Active Learning of Constraints for Semi-Supervised Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):43–54, 2013. [13](#), [95](#)
- J. Xu, A. G. Schwing, and R. Urtasun. Learning to Segment Under Various Forms of Weak Supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [95](#)
- J. Xu, J. Han, K. Xiong, and F. Nie. Robust and Sparse Fuzzy K-means Clustering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16*, pages 2224–2230, 2016. [67](#), [138](#)
- L. Xu. Bayesian Ying–Yang machine, clustering and number of clusters. *Pattern Recognition Letters*, 18(11):1167–1178, 1997. [25](#), [65](#), [124](#), [132](#)
- J. Yang, J. Liang, K. Wang, P. L. Rosin, and M.-H. Yang. Subspace Clustering via Good Neighbors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1537–1544, 2020. [10](#)
- M.-S. Yang and J. B. M. Benjamin. Feature-Weighted Possibilistic c-Means Clustering With a Feature-Reduction Framework. *IEEE Transactions on Fuzzy Systems*, 29(5):1093–1106, 2021. [12](#)
- M.-S. Yang and Y. Nataliani. Robust-learning fuzzy c-means clustering algorithm with unknown number of clusters. *Pattern Recognition*, 71:45 – 59, 2017. [74](#), [77](#)
- M.-S. Yang and H.-S. Tsai. A Gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction. *Pattern Recognition Letters*, 29(12):1713 – 1725, 2008. [40](#)
- Y. Yang. Dimensionality Reduced ℓ^0 -Sparse Subspace Clustering. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 2065–2074, 2018. [10](#)
- Y. Yao, Y. Li, B. Jiang, and H. Chen. Multiple Kernel k -Means Clustering by Selecting Representative Kernels. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2020. [8](#), [49](#), [105](#)
- T. Yellamraju and M. Boutin. Clusterability and Clustering of Images and Other “Real” High-Dimensional Data. *IEEE Transactions on Image Processing*, 27(4):1927–1938, 2018. [11](#)
- W. Ying, Y. Zhang, J. Huang, and Q. Yang. Transfer Learning via Learning to Transfer. In *International Conference on Machine Learning*, pages 5085–5094, 2018. [97](#)
- H. Yu, T. Zhang, Y. Lian, and Y. Cai. Co-regularized Multi-view Subspace Clustering. In *Proceedings of The 10th Asian Conference on Machine Learning*, pages 17–32, 2018. [10](#)
- J. Yu and M.-S. Yang. Optimality Test for Generalized FCM and Its Application to Parameter Selection. *IEEE Transactions on Fuzzy Systems*, 13(1):164–176, 2005. [41](#)

-
- J. Yu, Q. Cheng, and H. Huang. Analysis of the weighting exponent in the FCM. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):634–639, 2004. [74](#)
- Z. Yu, P. Luo, J. You, H.-S. Wong, H. Leung, S. Wu, J. Zhang, and G. Han. Incremental Semi-Supervised Clustering Ensemble for High Dimensional Data Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):701–714, 2015. [13](#), [95](#)
- Y. Yuan, D. Sun, and K. Toh. An Efficient Semismooth Newton Based Algorithm for Convex Clustering. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 5704–5712, 2018. [116](#)
- S. Zeng, X. Wang, X. Duan, S. Zeng, Z. Xiao, and D. Feng. Kernelized Mahalanobis Distance for Fuzzy Clustering. *IEEE Transactions on Fuzzy Systems*, pages 1–1, 2020. [12](#)
- Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang. A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review*, 2021. doi: <https://doi.org/10.1007/s10462-021-10042-y>. [78](#)
- B. Zhang. Generalized K-Harmonic Means - Boosting in Unsupervised Learning. *Hewlett-Packard Labs Technical Report HPL-2000-137*, 2000. [40](#)
- B. Zhang. Generalized K-Harmonic Means – Dynamic Weighting of Data in Unsupervised Learning. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–13, 2001. [40](#)
- B. Zhang, M. Hsu, and U. Dayal. K-Harmonic Means - A Data Clustering Algorithm. *Hewlett-Packard Labs Technical Report HPL-1999-124*, 1999. [39](#)
- C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, and D. Xu. Generalized Latent Multi-View Subspace Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):86–99, 2020a. [11](#)
- D.-Q. Zhang and S.-C. Chen. A Novel Kernelized Fuzzy C-means Algorithm with Application in Medical Image Segmentation. *Artificial Intelligence in Medicine*, 32(1):37–50, 2004. [40](#)
- L. Zhang, Q. Zhang, B. Du, D. Tao, and J. You. Robust Manifold Matrix Factorization for Joint Clustering and Feature Extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI’17*, page 1662–1668, 2017. [11](#)
- Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007. [75](#), [78](#), [79](#)
- Q. Zhang, W. Liu, and H. Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *2009 IEEE Congress on Evolutionary Computation*, pages 203–208, 2009. [75](#), [78](#), [80](#)
- R. Zhang and A. I. Rudnicky. A Large Scale Clustering Scheme for Kernel K-Means. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 4, pages 289–292, 2002. [40](#)

-
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM Sigmod*, volume 25, pages 103–114, 1996. [50](#)
- Y. Zhang and Y.-m. Cheung. An Ordinal Data Clustering Algorithm with Automated Distance Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6869–6876, 2020. [7](#)
- Y. Zhang, W. Wang, X. Zhang, and Y. Li. A cluster validity index for fuzzy clustering. *Information Sciences*, 178(4):1205–1218, 2008. [65](#), [132](#)
- Z. Zhang, K. Lange, and J. Xu. Simple and Scalable Sparse k-means Clustering via Feature Ranking. In *Advances in Neural Information Processing Systems*, volume 33, pages 10148–10160. Curran Associates, Inc., 2020b. [9](#)
- B. Zhao, J. T. Kwok, and C. Zhang. Multiple Kernel Clustering. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 638–649, 2009a. [95](#), [99](#)
- Q. Zhao, M. Xu, and P. Fränti. Sum-of-Squares Based Cluster Validity Index and Significance Analysis. In *International Conference on Adaptive and Natural Computing Algorithms*, volume 5495, pages 313–322. Springer, 2009b. [21](#), [25](#), [65](#), [124](#), [132](#)
- Y. Zhao, A. K. Shrivastava, and K. L. Tsui. Regularized Gaussian Mixture Model for High-Dimensional Clustering. *IEEE Transactions on Cybernetics*, 49(10):3677–3688, 2019. [10](#)
- X.-b. Zhi and J.-l. Fan. *Some Notes on K-Harmonic Means Clustering Algorithm*, pages 375–384. Springer Berlin Heidelberg, 2010. [40](#), [41](#), [43](#), [44](#)
- A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49, 2011. [75](#), [78](#)
- A. Zhou, Y. Wang, and J. Zhang. Objective extraction via fuzzy clustering in evolutionary many-objective optimization. *Information Sciences*, 509:343–355, 2020. [12](#)
- J. Zhou, W. Pedrycz, C. Gao, Z. Lai, J. Wan, and Z. Ming. Robust jointly sparse fuzzy clustering with neighborhood structure preservation. *IEEE Transactions on Fuzzy Systems*, pages 1–1, 2021a. [12](#)
- J. Zhou, W. Pedrycz, X. Yue, C. Gao, Z. Lai, and J. Wan. Projected fuzzy c-means clustering with locality preservation. *Pattern Recognition*, 113:107748, 2021b. [12](#)
- L. Zhou, X. Bai, D. Wang, X. Liu, J. Zhou, and E. Hancock. Latent Distribution Preserving Deep Subspace Clustering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4440–4446, 2019. [11](#)
- S. Zhou, X. Liu, M. Li, E. Zhu, L. Liu, C. Zhang, and J. Yin. Multiple Kernel Clustering With Neighbor-Kernel Subspace Segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 31(4):1351–1362, 2020. [8](#), [95](#), [99](#)
- Y. Zhou, Y. Zhu, Q. Ye, Q. Qiu, and J. Jiao. Weakly Supervised Instance Segmentation using Class Peak Response. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3800, 2018. [95](#)

- Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017. [95](#)
- X. Zhu, X. Liu, M. Li, E. Zhu, L. Liu, Z. Cai, J. Yin, and W. Gao. Localized Incomplete Multiple Kernel k -means. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18*, page 3271–3277, 2018. [8](#), [95](#), [99](#)
- F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. [97](#)