University of Technology, Sydney

Doctoral Thesis

Ensemble Rapid Centroid Estimation: A Semi-Stochastic Consensus Particle Swarm Approach for Large Scale Cluster Optimization

Author:
Mitchell Yuwono

Supervisors:
A/Prof. Steven W. Su
Dr. Bruce D. Moulton
Prof. Hung T. Nguyen

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

in the

Centre for Health Technologies
School of Electrical, Mechanical and Mechatronic Systems
Faculty of Engineering and Information Technology

February 2015

Declaration of Authorship

I, Mitchell Yuwono, declare that this thesis titled, 'Ensemble Rapid Centroid Estimation: A Semi-Stochastic Consensus Particle Swarm Approach for Large Scale Cluster Optimization' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Start by doing what's necessary; then do what's possible; and suddenly you are doing the impossible."

St. Francis of Assisi

UNIVERSITY OF TECHNOLOGY, SYDNEY

Abstract

Faculty of Engineering and Information Technology
School of Electrical, Mechanical and Mechatronic Systems

Doctor of Philosophy

Ensemble Rapid Centroid Estimation: A Semi-Stochastic Consensus Particle Swarm Approach for Large Scale Cluster Optimization

by Mitchell YUWONO

This thesis details rigorous theoretical and empirical analyses on the related works in the clustering literature based on the Particle Swarm Optimization (PSO) principles. In particular, we detail the discovery of disadvantages in Van Der Merwe - Engelbrecht's PSO clustering, Cohen - de Castro Particle Swarm Clustering (PSC), Szabo's modified PSC (mPSC) and Szabo's Fuzzy PSC (FPSC).

We validate, both theoretically and empirically, that Van Der Merwe - Engelbrecht's PSO clustering algorithm is not significantly better than the conventional k-means. We propose that under random initialization, the performance of their proposed algorithm diminishes exponentially as the number of classes or dimensions increase.

We unravel that the PSC, mPSC, and FPSC algorithms suffer from significant complexity issues which do not translate into performance. Their cognitive and social parameters have negligible effect to convergence and the algorithms generalize to the k-means, retaining all of its characteristics including the most severe: the curse of initial position. Furthermore we observe that the three algorithms, although proposed under varying names and time frames, behave similarly to the original PSC.

This thesis analyzes, both theoretically and empirically, the strengths and limitations of our proposed semi-stochastic particle swarm clustering algorithm, Rapid Centroid Estimation (RCE), self-evolutionary Ensemble RCE (ERCE), and Consensus Engrams, which are developed mainly to address the fundamental issues in PSO Clustering and the PSC families. The algorithms extend the scalability, applicability, and reliability of earlier approaches to handle large-scale non-convex cluster optimization in quasilinear complexity in both time and space. This thesis establishes the fundamentals, much surpassing those outlined in our published manuscripts.

Acknowledgements

First and foremost I would like to express my gratitude to my supervisors, A/Prof. Steven Su, Dr. Bruce D. Moulton, and Prof Hung T. Nguyen. Their vast knowledge, intelligence, expertise, understanding, and patience greatly expanded my knowledge and skill in many areas in the course of my PhD research. I really appreciate the considerable amount of efforts to help me with difficult research projects, critical reviews of my writings, and ultimately my scholarship applications. They are exceptional, inspirational, and excellent supervisors. Under their supervision, we publish high quality papers in prestigious journals.

A very special thanks goes to Prof. Hung T. Nguyen, who has been a visionary since the beginning of my Masters studies. Prof. Nguyen has inspired me to dive deeper into the world of biomedical technologies. It was under his supervision that I developed a focus and became interested in biomedical research and dared to explore the world of applied machine learning.

I would like to express my most sincere gratitude to my principal supervisor, A/Prof. Steven W. Su, who has been with me since the start of my PhD and has been through my ups and downs in my PhD life. He has seen through my struggles, supervised through numerous difficult research projects, and ultimately mend me to be an efficient, independent and resilient researcher. A/Prof. Steven is a kindhearted and understanding supervisor who is always supportive at all times.

I would also like to thank Dr. Bruce D. Moulton for being very critical on all aspects of my research life, especially with regards to the correctness of experiments, reports, and journals. His warm personality, deep knowledge and insights, and also attention to details are sincerely worth admiring. The works that I have done under his supervision have always been brought to perfection.

I would also like to thank to Dr. Jaime Valls Miro that has supervised me through the course of my Masters project in Advanced Robotics and supervised me during the summer research internship in ARC Centre for Autonomous Systems in UTS. The depth of his knowledge and his warm character have been a great inspiration throughout my life as a graduate student. Until today, Dr. Jaime has always been a role model in my life as a researcher.

I would also like to thank Dr. Ying Guo, Dr. Glenn Platt, Dr. Josh Wall, and Mr. Sam West from The Commonwealth Scientific and Industrial Research Organization (CSIRO) for the opportunity to apply our research in a practical industrial setting.

The members of the Centre for Health Technologies (CHT) have contributed to my personal and professional time at UTS. The group has been a source of friendships as well as good advice and collaboration. I am especially grateful to Father Ardi Handojoseno, who has been a source of inspiration, fun, moral and spiritual support during the course of my graduate studies, the up and downs in my Masters and PhD. I also would like to thank Phyo Phyo San; Henry Chandra; Rifai Chai; Andrew Zhang; Bao C. Q. Truong; Prof. Steve Ling; Ganesh Naik; Gibson Hu; Shuo Wang; Lei Zhang; and all of my colleagues whose name has not been mentioned here.

I would like to thank mom and dad for being the greatest parents in the universe. There is no words capable of conveying my feelings of gratitude towards the both of them. This thesis wouldn't exist if it is not because of their love. This thesis is my gift for them.

I would like to sincerely thank the love of my life. She has been there since the beginning of the writing until the completion of this thesis. Her contribution is enormous, both in this thesis and in my life as a whole.

Finally I would like to express my deepest gratitude to the sponsors for my scholarships including: University of Technology Sydney for their financial supports through the International Research Scholarship (IRS) and UTS President Scholarship (UTSP); The Commonwealth Scientific and Industrial Research Organization (CSIRO) for the scholarship and additional projects. I recognize that this research would not have been possible without their supports.

Mitchell Yuwono
University of Technology, Sydney
September 2014

Contents

List of Figures

List of Tables

Symbols

Variables and Constants

$a, b, c, n, v, w, x, y, z$	scalars, unless otherwise noted
K, C	number of clusters
$N, \lvert\mathbb{Y}\rvert$	number of data
$n_m, \lvert\mathbb{S}\rvert$	number of swarms
$n_i, \lvert\Theta\rvert$	number of particles
i, j, k, l, m, n	indices
d, dim	dimension of a feature vector, used interchangeably
t	time, iteration
T	period
f	function, frequency
$\mathbf{p}, \mathbf{g}, \mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z}$	vectors
$\mathbf{y}_j \in \mathbb{R}^{dim}$	the coordinate of the j^{th} observation vector
$\mathbf{x}_i, \mathbf{z}_i \in \mathbb{R}^{dim}$	the coordinate of the i^{th} voronoi cell
$\mathbf{v} \in \mathbb{R}^{dim}$	velocity vector
$\mathbf{so}, \mathbf{sc}, \mathbf{co}, \mathbf{mi} \in \mathbb{R}^{dim}$	self-organizing, social, cognitive, and local minimum vectors
$\mathbf{P}, \mathbf{G}, \mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$	matrices
$\mathbf{X}^M$,$\mathbb{X}^M$	local minimum memory matrix
$\mathbf{I}$	identity matrix
$\mathbb{S}, \mathbb{C}, \mathbb{V}$	sets
$\emptyset$	empty set
$\mathbf{u}, \mathbf{U}$,$\mathcal{U} \in \{0, 1\}$	fuzzy responsibility (membership) vector / matrix
$\mathsf{u}, \mathsf{U} \in [0, 1]$	crisp responsibility (membership) vector / matrix
π	3.141592653589793..., unless otherwise noted
Π	linear mapping kernel matrix
$\sigma, std(X)$	standard deviation
$\sigma^2, var(X)$	variance
$\Sigma, cov(X)$	covariance Matrix
ρ	Pearson's correlation

Λ, λ	Lagrange multiplier, eigenvalues
$\lambda_{(l)}$	the constant for the l^{th} term
ε, th	threshold
$\alpha, \beta, \gamma, \pi, \kappa, \delta, \tau$	various parameters
θ	multivariate Gaussian, particle
Θ	multivariate Gaussian Mixtures, swarm, subswarm
$\mathfrak{V}$	cluster validity index
$\mathcal{C}$	consensus matrix
φ	random number, random vector
Ω	search space
$\eta\%$	percentage
ω	inertia weight / velocity memory
$\psi_{j(l)} \in \mathbb{R}^{dim}$	j^{th} memory vector

Set Operations

$\sup(\mathbb{S})$	supremum (least upper bound) of a set
$\inf(\mathbb{S})$	infimum (greates lower bound) of a set
$\mathbb{R}$	set of real numbers
$\mathbb{R}^{dim}$	dim-dimensional vector of real numbers
$x \in \mathbb{X}$	x in $\mathbb{X}$
$x \notin \mathbb{X}$	x not in $\mathbb{X}$
$\lvert\mathbb{A}\rvert$	cardinality of a set
$\mathbb{A} \cup \mathbb{B}$	union of sets $\mathbb{A}$ and $\mathbb{B}$
$\mathbb{A} \cap \mathbb{B}$	intersection of sets $\mathbb{A}$ and $\mathbb{B}$
$\forall \mathbf{x} \in \mathbb{X}$	for each vector $\mathbf{x}$ in $\mathbb{X}$
$\exists \mathbf{x} \in \mathbb{X}$	there exists $\mathbf{x}$ in $\mathbb{X}$
$\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_K\}$	the clustered set $\mathbb{C}$, consisting of K clusters $\{\mathbb{C}_1, \ldots, \mathbb{C}_K\}$
$\mathbb{C}_x$	a voronoi region with coordinate of the voronoi cell defined in x
$y \in \mathbb{C}_x$	the observation y, crisply associated to $\mathbb{C}_x$.
$\mathbb{A}^c$	set complement
$\mathbb{B} \backslash \mathbb{A}$	relative complement of $\mathbb{A}$ in $\mathbb{B}$ such that $\mathbb{B} \backslash \mathbb{A} = \{x \in \mathbb{B} \lvert x \notin \mathbb{A}\}$

Matrix Operations

$\mathbf{X}^T$	matrix transpose
$\mathbf{XY}$	matrix multiplication
$\lvert\mathbf{X}\rvert$	matrix determinant
$adj(\mathbf{X})$	matrix adjoint
$tr(\mathbf{X})$, $Tr(\mathbf{X})$	matrix trace
$\mathbf{X}^{-1}$	matrix inverse

$\mathbf{X}^p$	matrix power
$\mathbf{X} \circ \mathbf{X}$	Hadamard product (elementwise multiplication)
$\mathbf{U}^T * \mathbf{U}$	Pairwise fuzzy T-norm operator
$diag(\mathbf{X})$	the diagonal components of $\mathbf{X}$
$diag(\mathbf{x})$	diagonal matrix which diagonal values are $\mathbf{x}$

Probability Functions

$p(x), q(x)$	probability distribution functions
$p(x\|z, \theta)$	conditional probability of x given z and the model θ
$p(x\|z, \Theta)$	conditional probability of x given z and the mixture model Θ
$p(x, y)$	joint probability of x and y
$E[X]$	expectation of X
$E[X\|Y]$	conditional expectation of X given Y
$\mathcal{L}(\theta)$	likelihood function
$H(X)$	Shannon's entropy of X
$H(X\|Y)$	conditional entropy of X given Y
$H(X, Y)$	joint entropy of X and Y
$I(X; Y)$	mutual information between X and Y
$NMI(X, Y)$	normalized mutual information between X and Y
$KL(P\|\|Q), D(P\|\|Q)$	Kullback-Leibler Divergence between P and Q
$D_{JSD}(P\|\|Q)$	Jensen-Shannon Divergence between P and Q
$\mathcal{P}(x)$	empirical probability distribution

General Functions

$\mu \pm \sigma$	average plus and minus standard deviation
$x > y$	x greater than y
$x \geq y$	x greater than or equal to y
$x < y$	x less than y
$x \leq y$	x less than or equal to y
$x \to \infty$	x approaches infinity
$x \leftarrow x + 2y$	assign $x + 2y$ as a new value for x
$\langle \mathbf{x}, \mathbf{y} \rangle$	inner product of $\mathbf{x}$ and $\mathbf{y}$
$\partial f(x, y)/\partial x$	partial derivative of $f(x, y)$ with respect to x
$\|x\|$	absolute value of x such that $\|x\| = x$ if $x \geq 0$ and $\|x\| = -x$ if $x < 0$
$\nabla_{\mathbf{z}} f(\mathbf{z})$	gradient of $f(\mathbf{z})$
$\log(x)$	natural logarithm of x, $\ln(x)$
$\exp(x)$	exponent of x, e^x
$\sum_{x \in \mathbb{X}} f(x)$	sum of $f(x)$ for all x in $\mathbb{X}$

$\frac{1}{\lvert\mathbb{X}\rvert}\sum_{x\in\mathbb{X}} f(x)$	average of $f(x)$ for all x in $\mathbb{X}$
$\frac{\sum_i w_i f(x_i)}{\sum_i w_i}$	weighted average of $f(x)$, indexed with i
$\prod_{x\in\mathbb{X}} f(x)$	product of $f(x)$ for all x in $\mathbb{X}$
$\phi(x)$	kernel function
$\mathcal{O}(N)$	complexity function
$y = \arg\max_x f(x)$	y is the x that maximizes $f(x)$
$y = \arg\min_x f(x)$	y is the x that minimizes $f(x)$
$d(\mathbf{x}, \mathbf{y}), d_{xy}$	distance between $\mathbf{x}$ and $\mathbf{y}$
$D(\mathbf{X}, \mathbf{Y})$	pairwise distance between vectors in $\mathbf{X}$ and $\mathbf{Y}$
$\Psi(\mathbf{x})$	resultant vector function for the voronoi cell $\mathbf{x}$

To Mom and Dad,

To Nina

List of Publications

1. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. An algorithm for scalable clustering: Ensemble rapid centroid estimation. *Conf Proc of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2014.

2. Mitchell Yuwono, Steven W. Su, Ying Guo, Bruce D. Moulton, and Hung T. Nguyen. Unsupervised nonparametric method for gait analysis using a waist-worn inertial sensor. *Applied Soft Computing*, 14, Part A: 72 – 80, 2014. ISSN 1568-4946. Special issue on hybrid intelligent methods for health technologies.

3. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Data clustering using variants of rapid centroid estimation. *IEEE Transactions on Evolutionary Computation*, 18: 3: 366–377, 2013.

4. Mitchell Yuwono, Steven W. Su, Ying Guo, Jiaming Li, Sam West and Josh Wall. Automatic Feature Selection Using Multiobjective Cluster Optimization for Fault Detection in a Heating Ventilation and Air Conditioning System. *Conf Proc IEEE International Conference on Artificial Intelligence, Modelling, and Simulation*, 2013: 1–6, 2013.

5. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Unsupervised segmentation of heel-strike IMU data using rapid cluster estimation of wavelet features. *Conf Proc IEEE Eng Med Biol Soc*, 2013: 953–956, 2013.

6. Mitchell Yuwono, Steven W Su, and Bruce D. Moulton. An Approach to Fall Detection using Gaussian Distribution of Clustered Knowledge. *Bio-Informatic Systems, Processing and Applications*. River Publishers. 2013: 69 – 82, 2013.

7. Mitchell Yuwono, Bruce D. Moulton, Steven W. Su, Branko G. Celler, and Hung T. Nguyen. Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *Biomedical Engineering Online*, 11: 9, 2012.

8. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Gait episode identification based on wavelet feature clustering of spectrogram images. *Conf Proc IEEE Eng Med Biol Soc*, 2012: 2949–2952, 2012.

9. Mitchell Yuwono, Jonathan Sepulveda, and A. M. Ardi Handojoseno. Centroid extraction from Hartmann-Shack images using swarm clustering approach. *Conf Proc IEEE Eng Med Biol Soc*, 2012: 1446–1449, 2012.

10. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Gait cycle spectrogram analysis using a torso-attached inertial sensor. *Conf Proc IEEE Eng Med Biol Soc*, 2012: 6539–6542, 2012.

11. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Fast unsupervised learning method for rapid estimation of cluster centroids. In *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, pages 889–896, June 10–15 2012.

12. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Optimization strategies for rapid centroid estimation. In *Proc. of the 34rd Annual International Conference of the IEEE EMBS*, pages 6212–6215, San Diego, Aug. 28–sept. 1 2012.

13. Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Method for increasing the computation speed of an unsupervised learning approach for data clustering. In *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, pages 2957–2964, June 10–15 2012.

14. Mitchell Yuwono, A. M. Handojoseno, and Hung T. Nguyen. Optimization of head movement recognition using Augmented Radial Basis Function Neural Network. *Conf Proc IEEE Eng Med Biol Soc*, 2011: 2776–2779, 2011.

15. Mitchell Yuwono. Unwrapping hartmann-shack images of off-axis aberration using artificial centroid injection method. In *IEEE Conference in Biomedical Engineering and Informatics (BMEI)*, pages 560–564, 2011.

16. Mitchell Yuwono, Steven W. Su, and Bruce D. Moulton. Fall detection using a Gaussian distribution of clustered knowledge, augmented radial basis neural-network, and multilayer perceptron. *Conf Proc 6th International Conference on Broadband and Biomedical Communications (IB2Com)*, 2011: 145–150, 21-24 Nov. 2011. doi: 10.1109/IB2Com.2011.6217909

Chapter 1

Introduction

Cluster analysis studies how systems can learn the representation of particular input patterns in a way that reflects the statistical structure of the overall collection [1]. It is exploratory in nature and often manifests in methods such as unsupervised learning, knowledge discovery, data mining, and pattern mining. Its objective is to discover valid, novel and potentially useful and ultimately understandable patterns in data [1–3]. Unlike classification, clustering algorithms assume no explicit target outputs, labeled responses, nor known evaluations [2, 4]. The decisions made are solely based on the structure of the input data based on a measure of similarity. Clustering methods are used in many practical applications in bioinformatics for example: Data mining for sequence analysis and genetic clustering [5–13]; Biomedical signal processing [14–23]; in medical imaging for image segmentation [24–35]; And object recognition [36–42].

This thesis is devoted for delivering the theoretical bases of our proposed Clustering Method, Ensemble Rapid Centroid Estimation (ERCE), alongside with its practical applications in various scenarios. This chapter gives an introductory remark on some of the key ingredients required to justify the contribution of our research. A brief overview on cluster analysis including various paradigms in Artificial Intelligence is covered in Section 1.1. A summary on the increasing trend for clustering methods in bioinformatics is presented in Section 1.2. A cursory view on a number of possible applications of clustering using our proposed algorithm is summarized in Section 1.3. A brief overview on the current challenges in clustering methodologies is presented in Section 1.4. Finally a brief highlight on the contribution of our method is summarized in Section 1.5.

1.1 Overview

Cluster analysis divides a data into groups (clusters) that are meaningful and possibly useful. Its main utility comes from its ability to extract and summarize the underlying statistical structure of the data. It has played important roles in solving many practical problems in a wide variety of fields. Among many others, bioinformatics is one of the fields that exploits the advancement of cluster analysis.

As a whole, clustering resides as a subgroup of Artificial Intelligence (AI). AI is generally defined as *an area of computer science that deals with giving machines the ability to seem like they have human intelligence* [43]. Merriam-Webster Online Dictionary defines "Learning" as *the activity or process of gaining knowledge or skill by studying, practicing, being taught, or experiencing something* [43]. In the context of Artificial Intelligence, the process of *Learning* describes how the behavior of an *unseen test dataset* can be predicted given a *known training dataset*.

Learning paradigms can be subdivided into three types [2]:

Definition 1.1.1 *(Supervised Learning/Classification).* In supervised learning (classification), a model is trained using data with known category labels. These labels are provided by the field experts [2]. An assumption made in this paradigm is that the provided label are true and valid. Optimal function approximation including training an Artificial Neural Networks (ANN) or decision trees is an NP-hard problem [44]. However, in supervised learning, the availability of a *teacher* greatly alleviate the overall *ambiguity* in decision making.

Definition 1.1.2 *(Unsupervised Learning/Clustering).* Contrast to Definition 1.1.1, the model in unsupervised learning (clustering) is trained using data with unknown category labels. The model is inferred solely based on the organization of the data itself. Because of this *unsupervised* nature, clustering is inherently harder than classification [2]. In general, clustering decision problem is NP-complete and the clustering optimization problem is NP-hard, except in some simple cases (e.g. constant number of clusters/clustering a 1-dimensional Euclidean problem) [45]. A solution which satisfies a given cluster optimality criterion does not necessarily guarantee an appropriate solution to the problem at hand. In higher dimensional datasets, where space between points become overwhelmingly sparse, clustering solutions can be arbitrarily far from optimal in the worst case [46].

Definition 1.1.3 *(Semi-supervised Learning/Semi-supervised clustering).* Semi supervised clustering incorporates the concept in Definition 1.1.1 into Definition 1.1.2. In semi- supervised clustering, instead of specifying the class labels, pair-wise constraints (e.g. *must-link*: Two objects must reside in the same clusters; or *cannot-link*: Two objects must reside in different clusters;) are specified.

These constraint provides a weak encoding of knowledge that can be beneficial for data clustering, where precise definitions of underlying clusters are absent [2]. Even though semi-supervised clustering is likewise an NP-hard problem, the presence of these constraints alleviate some of the inherent difficulty in clustering.

1.2 Clustering in Biomedical Informatics

The amount of stored information, particularly in the field of Biomedical Engineering and Bioinformatics, has been growing exponentially. For example, in genome sequence studies alone, Berman et al. estimated that the human's genome DNA would contain around a Gigabyte of information. In order to process the 100,000 translated proteins and 32,000,000 amino acids, Berman et al. approximates a total memory requirement of up to the order of 200 Gigabytes [47]. Medicine and biomedical sciences have also become increasingly data-intensive. Emerging medical technologies require sophisticated mathematics, signal processing, artificial intelligence, data analysis and inference systems. Such higher level of sophistication requires abundant information from various natural and electrical sources. The current era, for all these reasons, can be considered as a golden age for Biomedical Informatics [48].

The number of publications indexed in PubMed under the topic of 'Clustering' alone has grown from 1021 (0.44% of total indexed publications) in 1990 to 11289 (13.20% of total indexed publications) in 2013 [49]. Publications under the topic of 'Data Mining' has grown from 35 (0.02% of total indexed publications) in 1990 to 1703 (1.99% of total indexed publications) in 2013. Publications under the topic of 'Knowledge Discovery' grown from 46 (0.02% of total indexed publications) to 1021 (1.19% of total indexed publications) in 2013. The stacked bar graph showing the total publications under the three topics is shown in Figure 1.1.

1.3 Applications

Clustering has become the basic and crucial component of the day-to-day activities of researchers, scientists, clinicians, nurses and decision-makers [48]. To date clustering has been used mainly for three main purposes [2]:

1. *Discover underlying structure*: to gain insight into data, generate hypotheses, detect anomalies, and identify salient features.

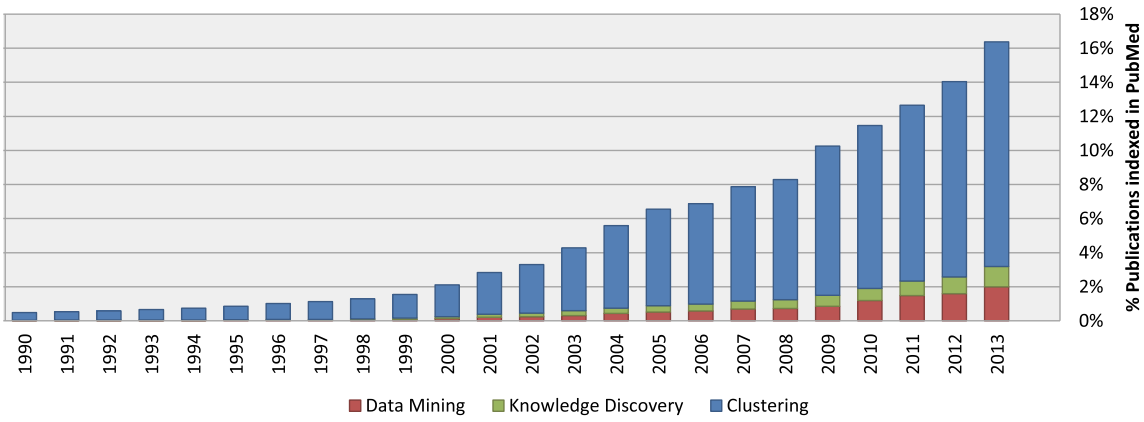


FIGURE 1.1: % publications indexed in PubMed from 1990 to 2013 on the specific keywords/-topics: Clustering, Data Mining, and Knowledge Discovery [49].

2. *Natural classification*: to identify the degree of similarity among forms or organisms (phylogenetic relationship).

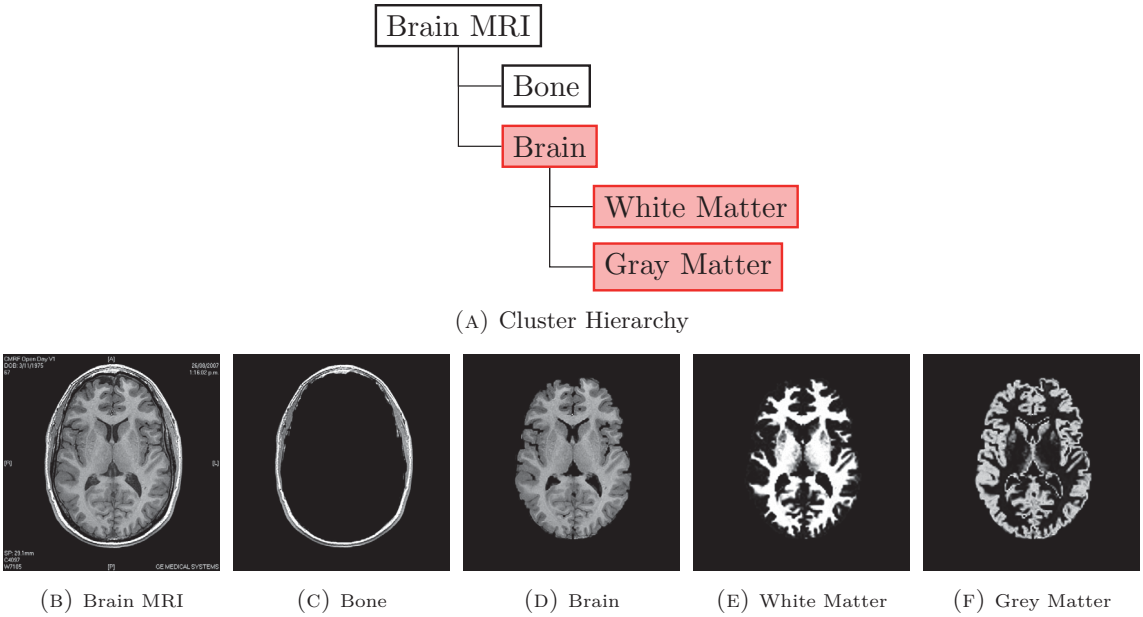
3. *Compression*: as a method for organizing the data and summarizing it through cluster prototypes.

Concrete examples are seen in biomedical image processing [24–35]. Clustering techniques are useful for applications such as Magnetic Resonance Image (MRI) segmentation [29, 50], adaptive texture segmentation for tissue and cell recognition [27, 28, 31, 51], image analysis of Hartmann-Shack Wavefront sensor [24, 25], gait signal and spectrogram segmentation [16, 22], and retinal fundus image segmentation [52–54].

Robust segmentation of skeleton and brain segment — including white and grey matter — can be done using Rapid Centroid Estimation (RCE) [55]. A sample segmentation can be seen in Figure 1.2.

Figure 1.3 shows the usage of clustering for wavefront reconstruction in Hartmann-Shack (HS) Aberrometry. Wavefront aberrometry has revolutionized the measurement of lower order and higher order aberrations, including astigmatism, defocus, spherical, coma, trefoil, pentafoil, hexafoil, and tetrafoil to name a few. In two of our published manuscript, we use cluster analysis to automate the unwrapping process of noise-ridden HS images and "injecting" artificial foci to compensate for missing and occluded ones [24, 25].

Clustering can also be applied to retinal photography. Retinal photography is useful for the early detection of diabetic changes, hypertensive retinopathy, macular degeneration, optic nerve disease, and retinal holes or thinning [56]. We applied our proposed algorithm to perform segmentation of



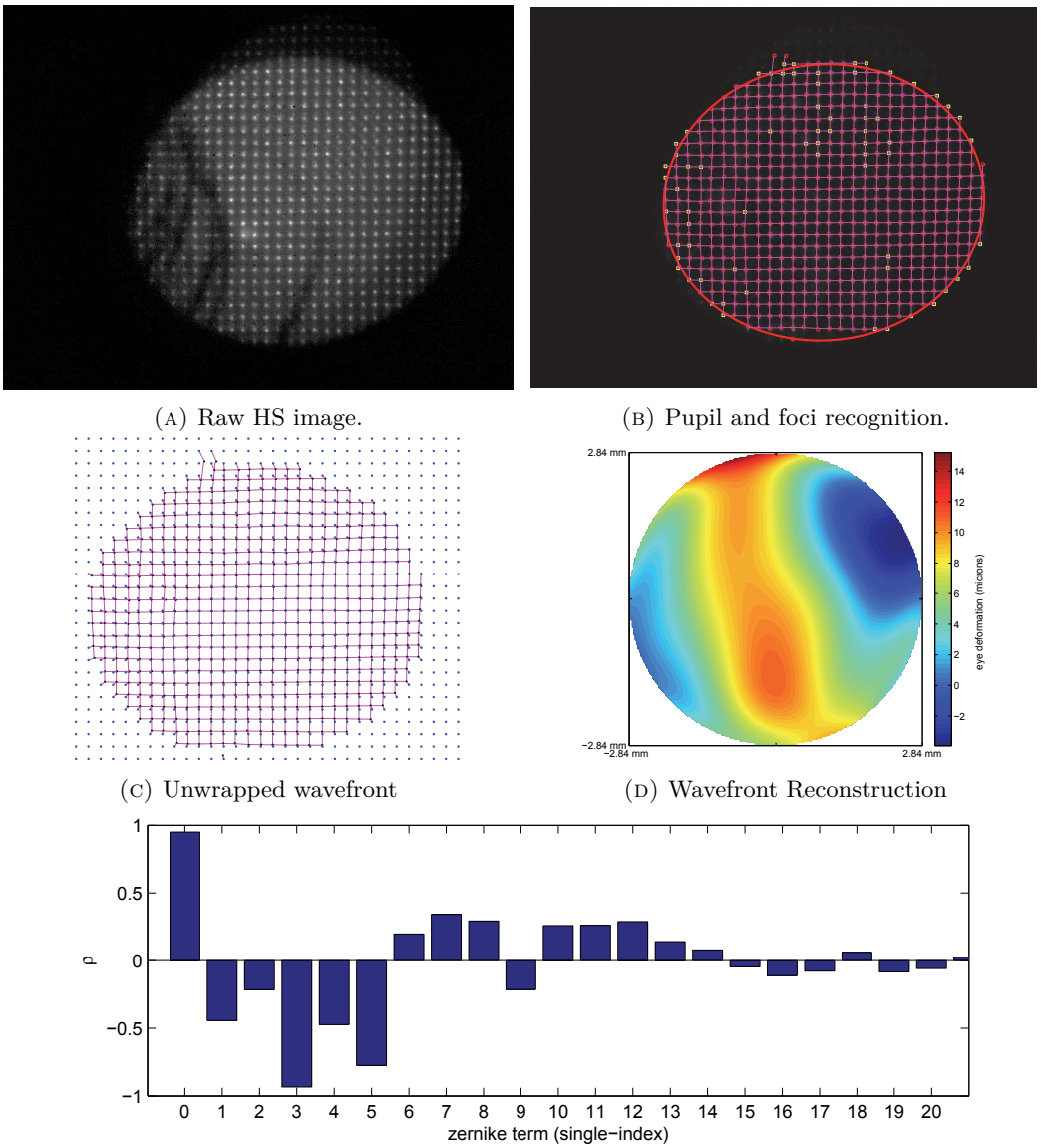
(A) Cluster Hierarchy

(B) Brain MRI (C) Bone (D) Brain (E) White Matter (F) Grey Matter

FIGURE 1.2: Clustering a Magnetic Resonance Image. The clustering algorithm uses Rapid Centroid Estimation (RCE) [4].

the optic disc, macula, and blood vessels from retinal fundus images. On this particular application, we used an ensemble variant of our proposed clustering algorithm: Ensemble Rapid Centroid Estimation (ERCE) together with morphological filtering techniques including line detector [57] and geodesic morphology operations [53]. Some of the results can be seen in Figure 1.4.

Clustering has also been used in various biomedical signal processing applications including gait analysis [16, 17, 22]. Poor gait quality is highly correlated with the increase in frequency of falling among older adults [58]. The important gait parameters for calculating the measures includes: the bilateral step length; the precision timing of the bilateral heel-strike events; and the precision timing of the bilateral toe-off events [17, 58]. Figure 1.5 shows the overall result of the proposed method. The first three principal components of the accelerometric and gyroscopic wavelet coefficients can be seen in Figure 1.5b. This example shows a 3 seconds data snippet from a 6.7 minutes normal gait data (24229 observations). A total of 690 heel strike and toe-off events or approximately 345 strides were detected. Detailed results on gait analysis and fall detection can be seen in our published papers [14, 16, 17, 22, 23, 59].



(A) Raw HS image.

(B) Pupil and foci recognition.

(C) Unwrapped wavefront

(D) Wavefront Reconstruction

(E) Estimated Zernike terms (single index).

FIGURE 1.3: Reconstructing Hartmann Shack Centroids using RCE and Artificial Centroid Injection [24, 25].

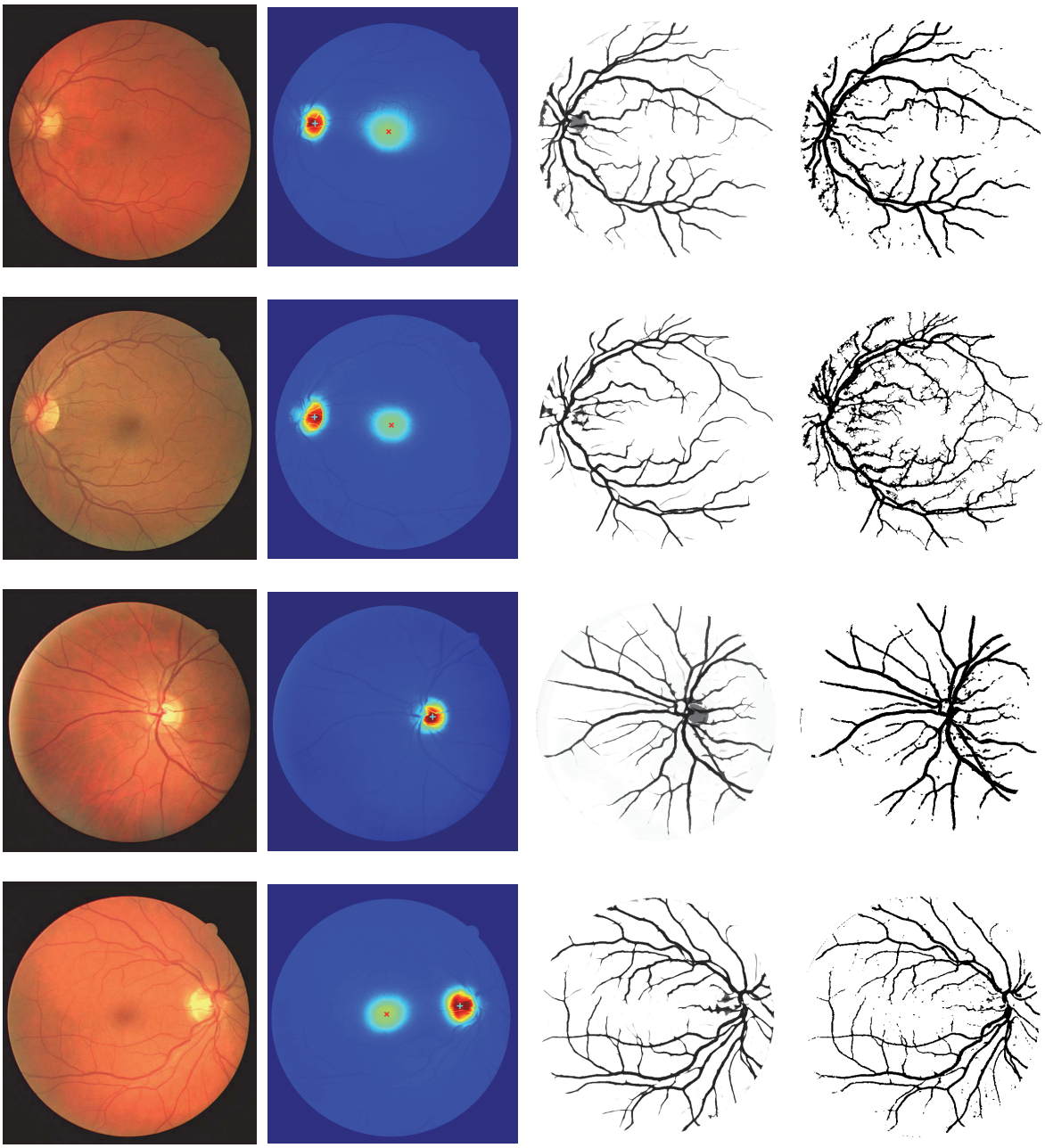
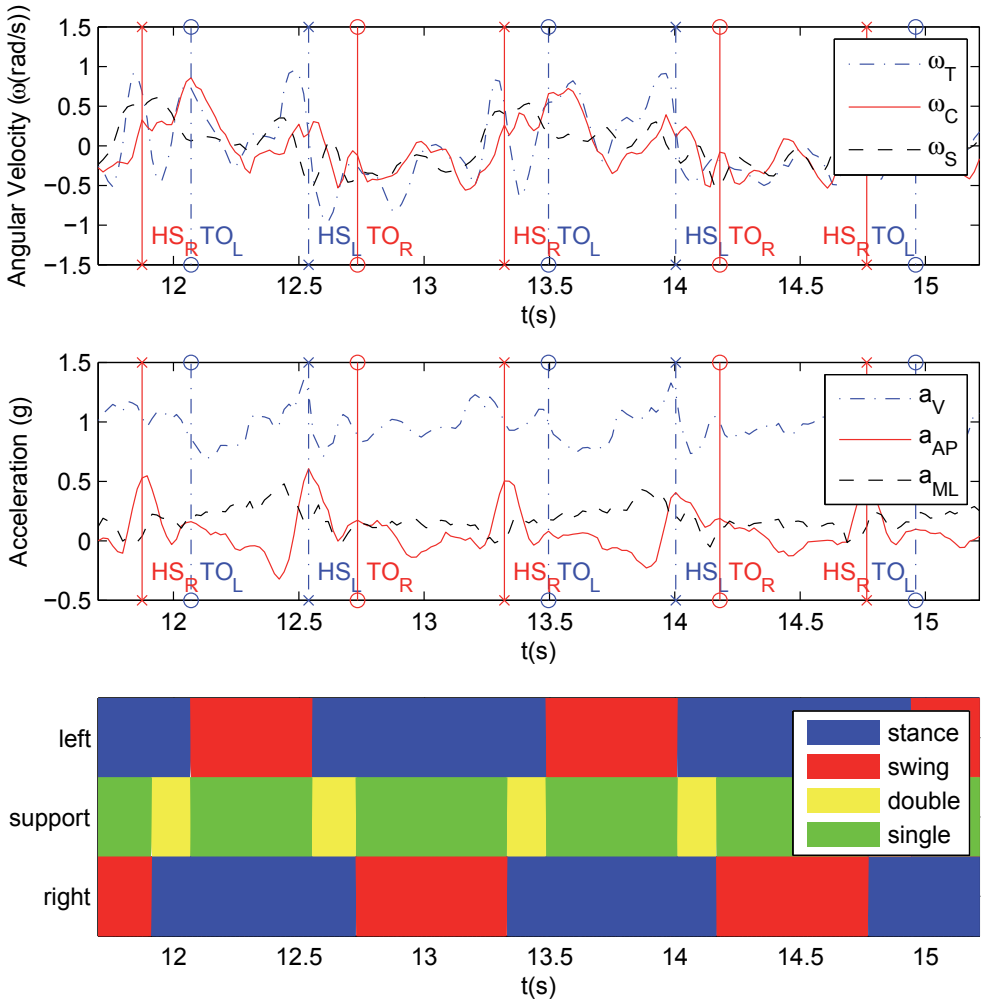
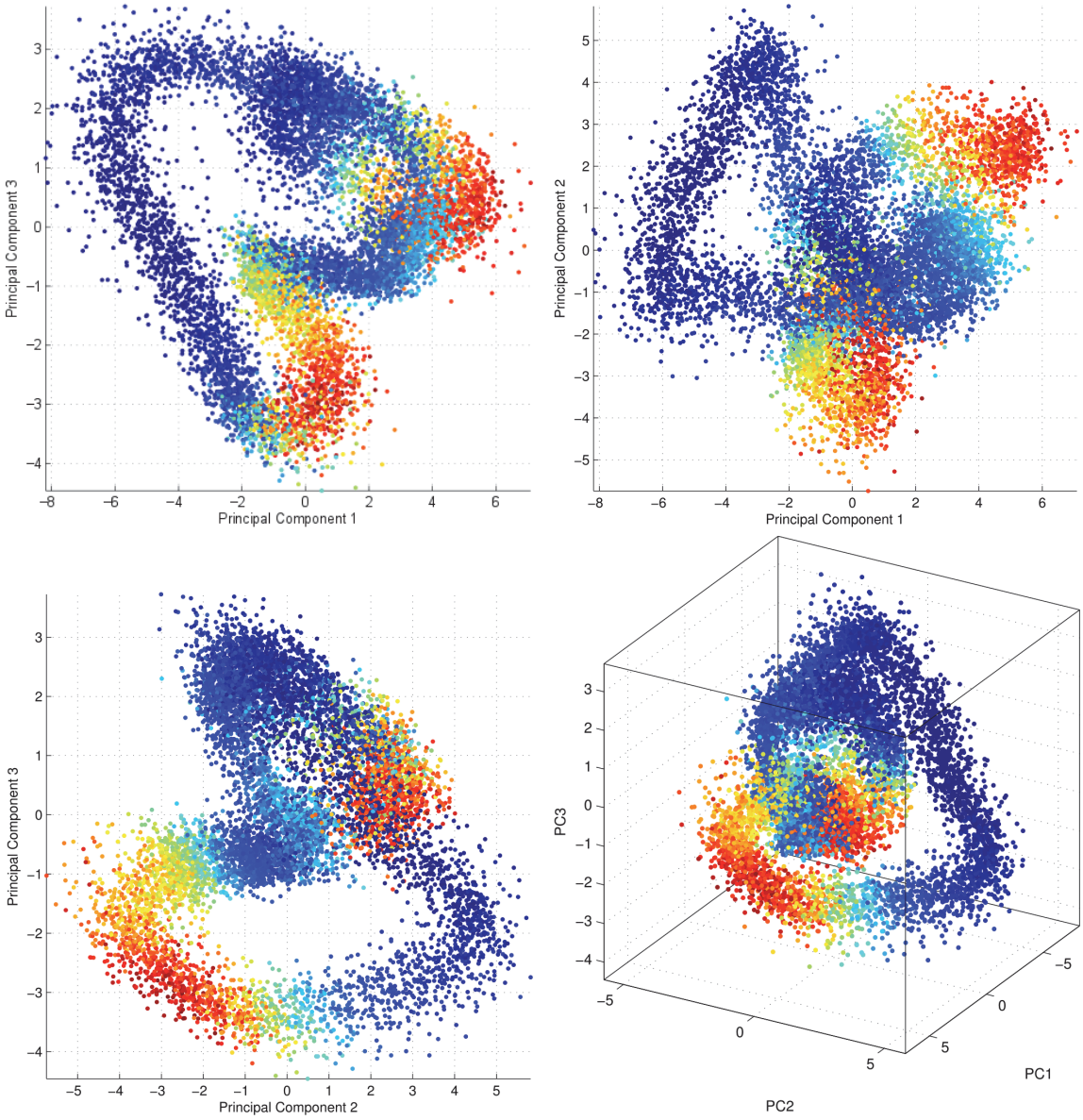


FIGURE 1.4: Clustering retinal fundus images using ERCE [55]. Left to right: {raw retinal fundus image}; {segmented optical disc, macula, and fovea}; {segmented retinal blood vessels}; and {classified detection mask}.



(A) ERCE clustering result on the gyroscopic and accelerometric wavelet features. The important gait parameters including the bilateral step length; the precision timing of the bilateral heel-strike events; the precision timing of the bilateral toe-off events; and the overall gait phase can be extracted. With this approach medical practitioners and kinesiologists can objectively examine the evidence for assessing fall risk in an individual.

FIGURE 1.5: Estimating gait parameters using ERCE [17, 55] (cont.).



(B) The first three principal components of the accelerometric and gyroscopic wavelet coefficients obtained after Principal Component Analysis. The cyclical character of the human gait is clearly pronounced in the Eigenspace. Scatter points with hot (light yellow – dark red) color mask indicate higher likelihood of positive Heel-Strike detections.

FIGURE 1.5: Estimating gait parameters using ERCE [17, 55].

1.4 Challenges

While the application of clustering may be straightforward, the underlying process behind clustering is very challenging [2]. Clustering is inherently an ill-posed problem where the goal is to partition the data into some unknown number of clusters based on intrinsic information alone [2]. This data-driven nature makes it very difficult to design an algorithm that can correctly and efficiently discover the natural clusters in the given data.

Divisive clustering such as the k-means algorithm is frequently used for large scale data clustering because of its low time and memory complexity ($\mathcal{O}(N)$). However its formulation limits k-means to solving only globular/gaussian clusters. The number of k has to be pre-specified, it also does not impose any hierarchical tree which makes its result harder to interpret than hierarchical agglomerative clustering. Further, specifying different k values greatly affects its result. The k-means is also sensitive to outliers and different initialization values.

The hierarchical agglomerative clustering [60], in the other hand, does not have any k-means limitations because it does not assume the number to be known beforehand. By specifying the linkage criterion, agglomerative algorithms can readily accommodate any cluster shape. The hierarchical tree makes the results of agglomerative clustering easier to interpret. However, due to its quadratic time and memory complexity ($\mathcal{O}(N^2)$), agglomerative clustering algorithms do not scale well to larger dataset. Other clustering algorithms that uses proximity matrix such as spectral clustering [61], Multi-Objective Clustering with Automatic K-determination (MOCK) [62], Density-based spatial clustering of applications with noise (DBSCAN) [63], and Evidence Accumulation clustering ensembles [64–66], also encounter similar scalability issue.

A good clustering algorithm needs to satisfy a number of requirements such as: scalability; flexibility; capability to discover clusters with arbitrary shapes; adaptiveness; robustness to noise and outliers; insensitivity to the order of input records; and simplicity of interpretation. The currently available clustering techniques do not address all the requirements adequately and concurrently.

Several attempts proposed in the previous literature to alleviate these problems of k-means clustering include cluster validation schemes [67], entropy weighting [68], information theoretic measures[69], stochastic methods [4, 70–75] and ensemble schemes [64, 76]. Ensemble algorithms are powerful alternatives to clustering because they combine the strength of both divisive and agglomerative paradigms. The clusters created with ensemble algorithms tend to be highly intuitive. However, ensemble algorithms suffer from scalability issue on larger datasets.

1.5 Contribution of This Thesis

The major contributions of this thesis can be summarized as follows.

- **Proposition of Rapid Centroid Estimation**

  Rapid Centroid Estimation (RCE) [4] is a light-weight semi-stochastic clustering algorithm which is inspired by Eberhart and Kennedy's Particle Swarm Optimization (PSO) [77] and Cohen and de Castro Particle Swarm Clustering (PSC) [75]. We introduced the algorithm in 2012 [71, 73] as a lightweight simplification to the PSC algorithm [75]. Further strategies to improve its optimization capability are proposed in our subsequent publications [4, 72]. RCE shares similarly to its deterministic predecessor (k-means algorithm) that it clusters data using the voronoi cells principle. Due to its semi-stochastic nature, the RCE is almost guaranteed to discover the natural local optimum solution given the correct number of clusters, parameters, and sufficient number iterations are supplied. The particle trajectory of RCE can be seen in Figure 1.6.

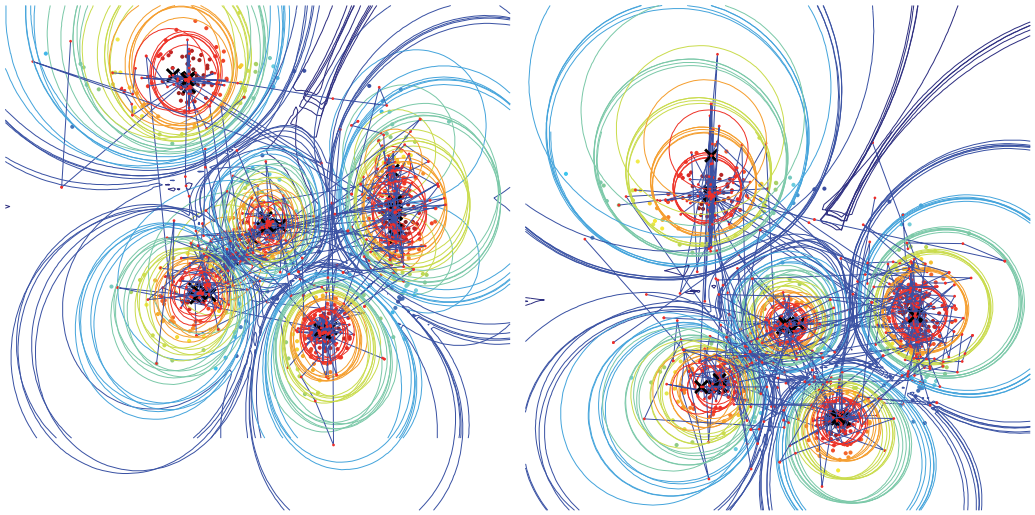


FIGURE 1.6: Trajectory of the *swarm*{6} RCE^{r+} 2014 particles (randomly seeded initial positions) recorded after 30 iterations on the five Gaussian dataset using both *substitution* and *particle reset* strategies shows *Swarm*{6} RCE^{r+} robustness and insensitivity to initialization.

- **Proposition of Ensemble Rapid Centroid Estimation (ERCE)**

  Ensemble RCE (ERCE) [55] was proposed in 2014 as an ensemble extension to RCE, improving RCE in terms of memory and computational efficiency, capability to automatically determine the number of clusters, and gracefully handle non-convex datasets in quasilinear complexity. Using the self-evolving cooperative semi-stochastic swarm ERCE constructs the

essential building blocks required for large scale consensus clustering using minimal memory and computational resources. Benefiting from the robustness of swarm intelligence, the versatility of voronoi tessellation and the flexibility of graph algorithms, the ERCE is designed to discover natural groupings in both convex and non-convex data. Some results on non-convex data can be seen in Figure 1.7. The visual abstract, giving the brief visual overview of the ERCE, can be seen in Figure 1.8.

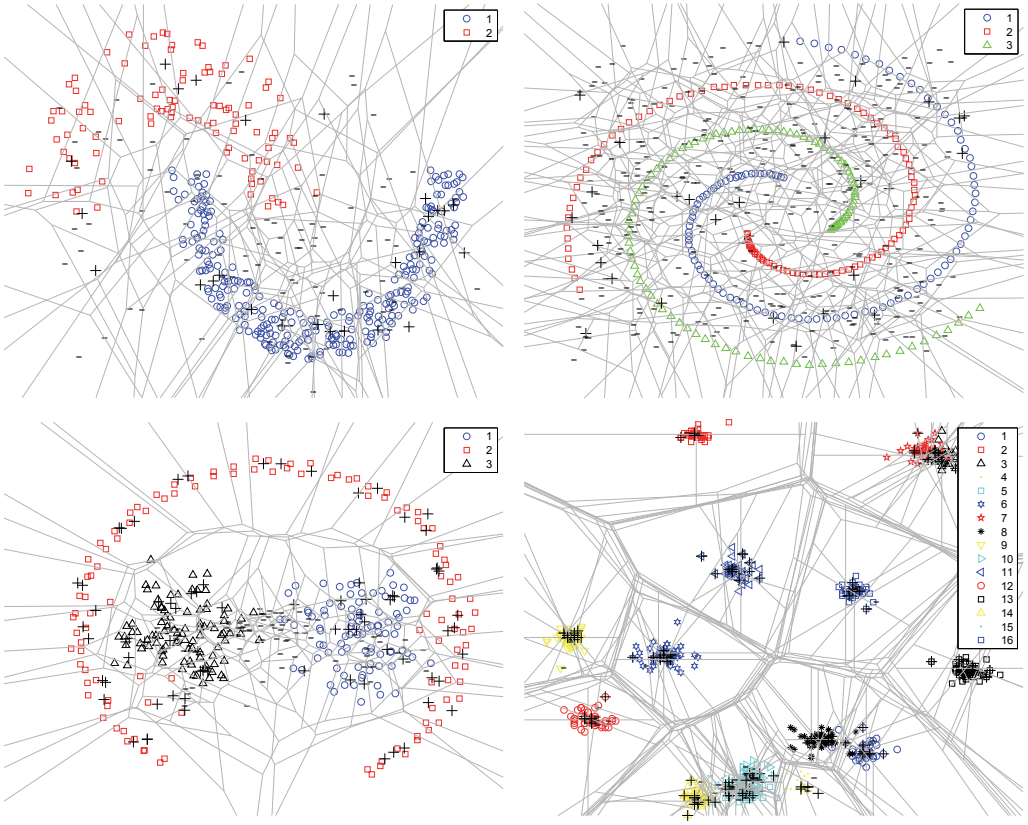
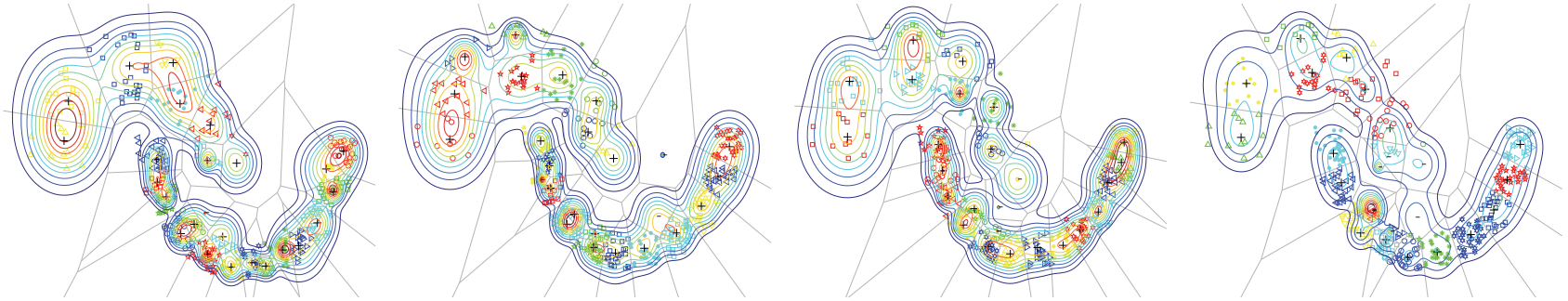


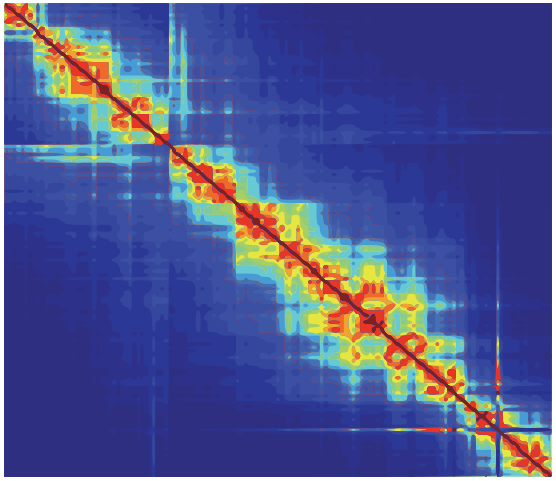
FIGURE 1.7: Clustering Non-convex dataset using ERCE. + and - signs denotes positive and negatively charged particles, respectively.

- **Consensus Engrams**

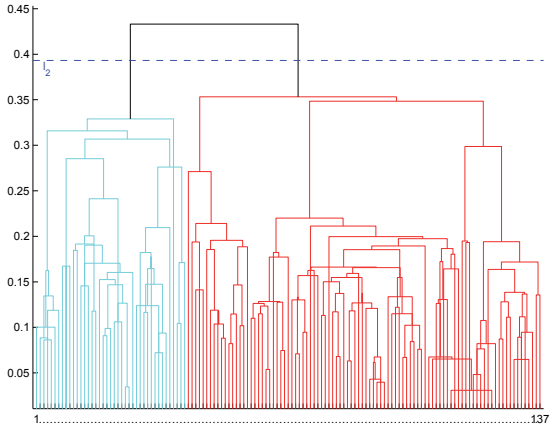
  Inspired by Hebb's theories on cell assembly [78], we arrange the particles in the swarm in a fully connected construction called the consensus engrams. This approach substantially increases the scalability of the swarm to handle data whose total size exceeds the physical random access memory (RAM) limit of the machine. The method is specifically designed to extract statistical structure from incomplete data of large volume and noise content. It is insensitive to neither the volume nor the order of data and the engrams only need to "see"



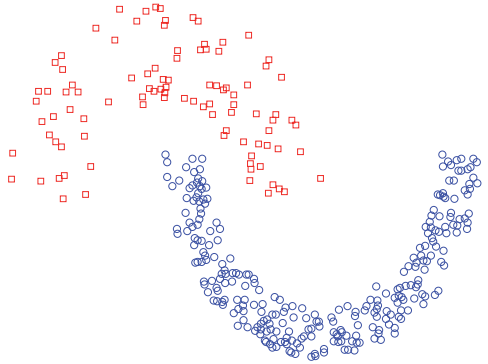
(A) Various fuzzy representations of the data obtained using ERCE. Each swarm in ERCE produces non-identical voronoi tessellations on the data. The degree of fuzziness for each cluster is then optimized using the tradeoff between cluster entropy and the degree of fuzzified dissimilarity [55].



(B) The compressed co-association matrix obtained using the CA-tree [66] and fuzzy WEAC [65, 76] Hybrid [55].



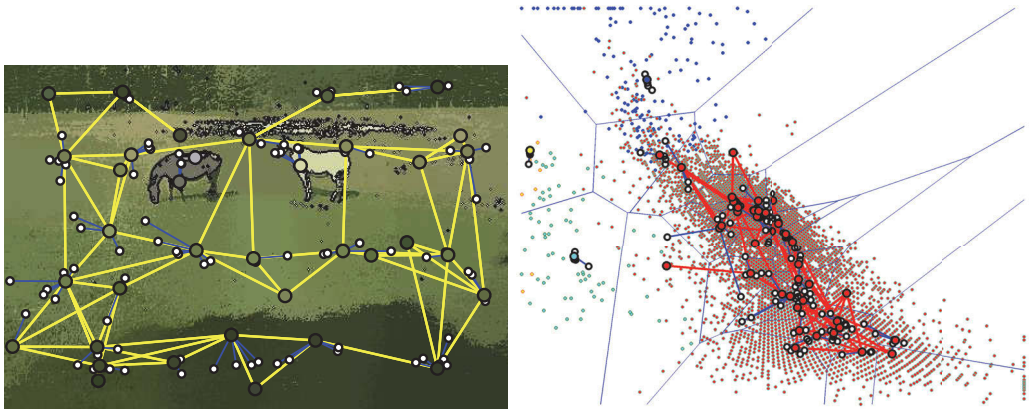
(C) The optimum cut for the corresponding CA-tree using the highest lifetime criterion [64].



(D) Natural grouping is recovered by performing inverse mapping on the CA-tree [55, 66].

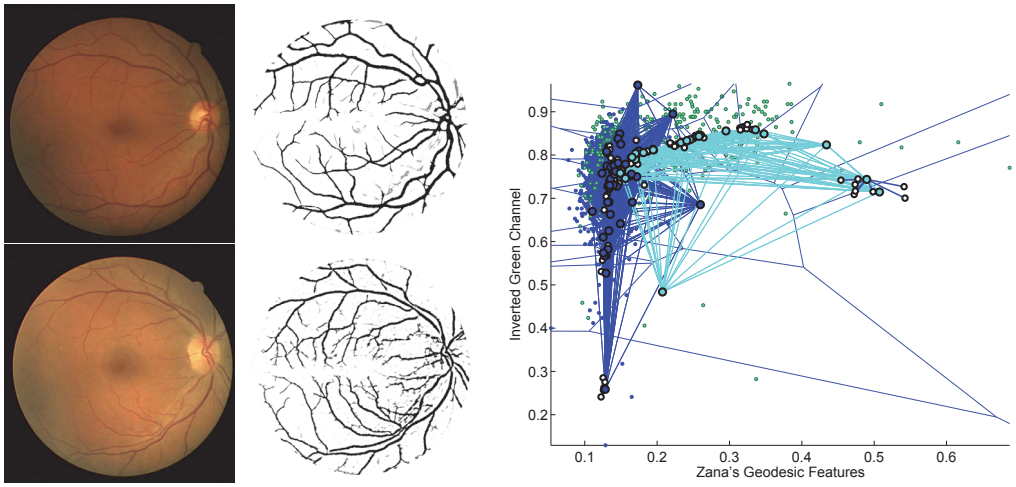
FIGURE 1.8: The algorithmic flow of ERCE explained using an artificial dataset [55].

just enough representative data to learn the overall statistical structure. A sample run of the algorithm on a high resolution image segmentation problem is shown in Figure 1.9.



(A) A color image segmentation result and the Consensus engrams overlay. Image source: Lotus Hill Institute [79].

(B) Scatter plot of the color image (HSV) feature space and the consensus engrams encoding. x: saturation; y: value.



(C) Blood vessel extraction from retinal fundus images. Image source: DRIVE [80].

(D) Scatter plot of the retinal fundus feature space and the consensus engrams encoding.

FIGURE 1.9: Various Segmentation results using ERCE (Consensus Engrams). Further details can be seen in Chapter 6.

- **Stability conditions of Particle Swarm Optimization (PSO)**

  In Lemma 3.1.1 (Convergence of PSO), we revisit the convergence and stability conditions of Particle Swarm Optimization (PSO) from control theory point of view. Lemma 3.1.1 reveals that the particle swarm optimization can be generally modeled as a second order linear system. Particularly, we discover that Jiang et al.'s stability boundary [81] is incorrectly derived. The final value at convergence agrees with the earlier research [81, 82].

- **Suboptimal Convergence of Van Der Merwe - Engelbrecht's PSO Clustering [83]**

In Section 3.3 (PSO trajectory suboptimality), we discover a significant issue in Van Der Merwe - Engelbrecht's proposition [83]. We reveal that *unless at least one of the particles is pre-initialized near to the optimum, the probability of the swarm to converge to the maximum likelihood coordinates degrades exponentially as the number of classes or dimensions of the clustering problem increase.*

- **Complexity Issues in Cohen and de Castro's PSC [75]**

  In Section 4.3 (PSC — Complexity Analysis), we analyze the theoretical and empirical computational and memory complexity of PSC. This chapter details important efficiency issues of the algorithm which restricts its applicability to only smaller datasets. For example, with a 400x1000 data matrix, on each iteration the PSC already suffers from a significant 2000% slow-down compared to k-means. In order to cluster 1000 observations of 1000 dimensional double precision data into 256 clusters, the algorithm variants require as much as 2GB memory; whereas k-means, fuzzy c-means, and RCE require memory allocation of lower than 15MB. Ironically, empirical results on benchmark data from [84] presented in Table 4.3, Table 4.5, and Table 4.7 show that these high complexities do not generally translate into any increase in cluster quality.

- **Stability of Cohen and de Castro's PSC [75] and sensitivity to initialization**

  In Theorem 4.1 (PSC's resemblance to k-means), we derive the stability criteria of the PSC algorithm and a detailed proof of convergence. The proof reveals a problematic cyclical trajectory of the particles using parameter selection outlined in the standard PSO (Lemma 3.1.1). Interestingly this crucial character is discussed neither in the initial manuscript [75] nor its follow ups [70, 74, 85].

  When the convergence bounds described in Equation 4.37 are properly satisfied, an even more interesting phenomenon is observed. PSC guarantees convergence as long as the self-organizing constant is nonzero. Furthermore, both social and cognitive parameters have negligible effect to the swarm's convergence, which implies that these parameters are redundant. The ultimate implication of this theorem is that the PSC generalizes to the k-means, retaining all of its performance aspects including the most severe: the curse of initial position.

  In 2010, Szabo et al. [85] observed this particular convergence phenomenon in their empirical experiment [85]. They suggested that *"there were no much gain obtained with the PSC when compared with a standard self-organizing clustering methods"* [85]. However, the reason behind the phenomenon were not further explained. Theorem 4.1 therefore provides the theoretical proof necessary to complete their analysis.

- **Particle behavior of the PSC Families**

In the empirical testing following the theoretical proof of Theorem 4.1 (PSC's resemblance to k-means), we discover that all PSC families, including the original Cohen - de Castro's PSC [75], Szabo's Modified memoryless Particle Swarm Clustering (mPSC) [74], and Szabo's Fuzzy Particle Swarm Clustering (FPSC) [70], share the exact particle behavior. The three algorithms, although proposed under varying names and time frames, are conceptual duplicates of the original PSC.

- **RCE and simplification of the PSC**

  In Definition 5.2.10 (Resultant Vector) we propose a simplified update rule for PSC derived based on the proof in Theorem 4.1. By calculating only the resultant vector, the time complexity for each position update can be reduced up to as low as that of the k-means. Concurrently, this new redefinitions reveals two problems associated with the PSC original formulation: The first problem is concerned with computational complexity of random number generation; The second is concerned with the dilution of randomness as a consequence of Kolmogorov's strong law of large number. This discovery leads to the proposition of Definition 5.2.11 (Simplified Resultant Vector) — the latter is used in RCE — which addresses these issues.

- **RCE Strategies: Substitution, Particle Reset, and Multi-Swarm**

  In Theorem 5.1 (RCE's resemblance to k-means) we prove that the RCE in its fundamental form inherits the consequence of Theorem 4.1 (PSC's resemblance to k-means) including the curse of initial position. Conceding this fact, we propose strategies including substitution, particle reset, and multi-swarm in order to alleviate the severity of the curse. Especially when the RCE is operated as a multi-swarm, each of the RCE subswarm can be assigned partial random sampled data which greatly reduces computational burden and increases swarm diversity. These characteristics are particularly desirable when the swarm are to be aggregated using consensus/ensemble clustering.

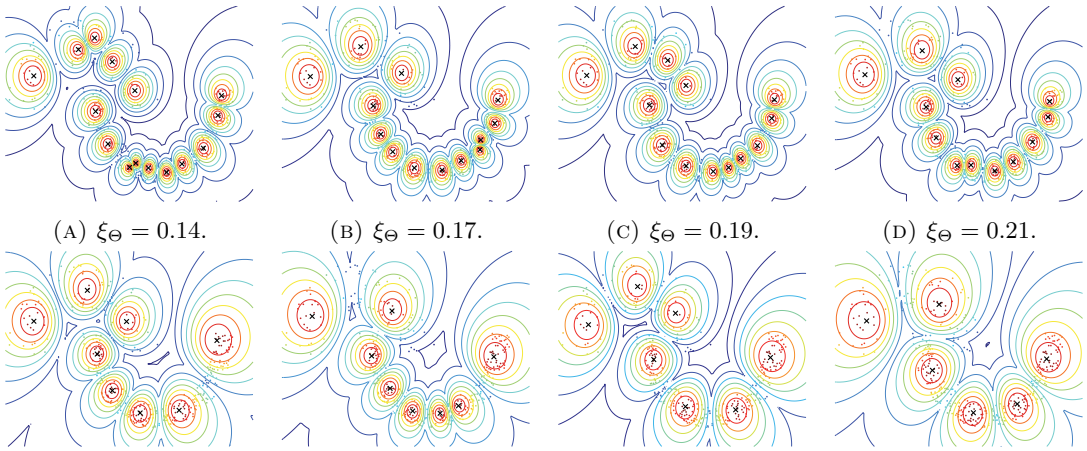
- **ERCE: Charged Particles**

  The presence of redundant partitions in ensemble clustering aggregation may produce an undesireable bias to the final solution [86]. In order to diversify the particles, we endow them with either positive or negative charge. As a result, a constant chaotic turbulence between particles is created in the search space such that the probability of creating a duplicate partition is minimized. The approach is outlined in Section 6.1.1.

- **ERCE: Self-evolution**

  The self-evolving swarm RCE equips each subswarm with the ability to summon additional particles until the average cluster entropy criterion ξ [87] is minimized. According to Bezdek,

minimizing the cluster entropy is equivalent to maximizing the information content in each cluster [87]. The resulting ensemble partitions using the self-evolution scheme can be seen in Figure 1.10.



(A) $\xi_\Theta = 0.14$. (B) $\xi_\Theta = 0.17$. (C) $\xi_\Theta = 0.19$. (D) $\xi_\Theta = 0.21$.

(E) $\xi_\Theta = 0.23$. (F) $\xi_\Theta = 0.26$. (G) $\xi_\Theta = 0.28$. (H) $\xi_\Theta = 0.3$.

FIGURE 1.10: Variability of K due to ξ in ERCE swarms on the Half Rings dataset.

- **ERCE: Calculating Consensus Clustering in Quasilinear Time and Space**

  The ERCE ensemble aggregation method uses the CA-tree which allows it to compress redundant label vectors and calculate the consensus matrix in quasilinear complexity. Using the compression map obtained from the CA-tree, the fuzzy consensus can be obtained from the compressed fuzzy membership matrices. ERCE achieves the lowest memory consumption compared to other conventional consensus techniques as shown in Figure 1.11.

- **ERCE + Consensus Engrams: Theoretical Foundations and Preliminary Empirical Results on Color Image Segmentation and Blood Vessel Segmentation from Retinal Fundus Images.**

  In neuropsychology, engrams are means by which memory traces are stored as biophysical or biochemical changes in the brain (and other neural tissues) in response to various external stimuli which leave a lasting trace in the brain [78]. Inspired by Hebb's theorem, we engrave the essential fundamentals required to encode a statistical structure inside the ERCE swarm as detailed in Section 6.3. The definitions are mainly derived based on information theory according to Kullback and Leibler [88], and Shannon [89].

  Based on the empirical results on color image segmentation problems (Figure 1.12), we observe that the proposed method achieved linear complexity in both time and space compared to Shi-Malik's normalized cuts [90] and conventional consensus clustering [91].

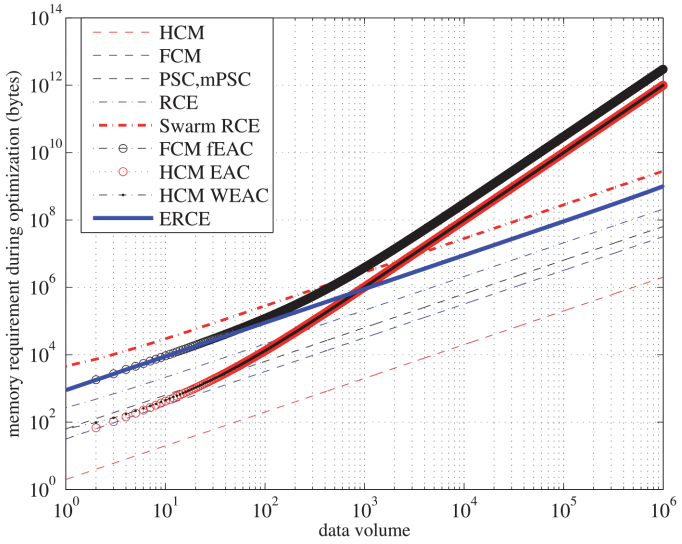
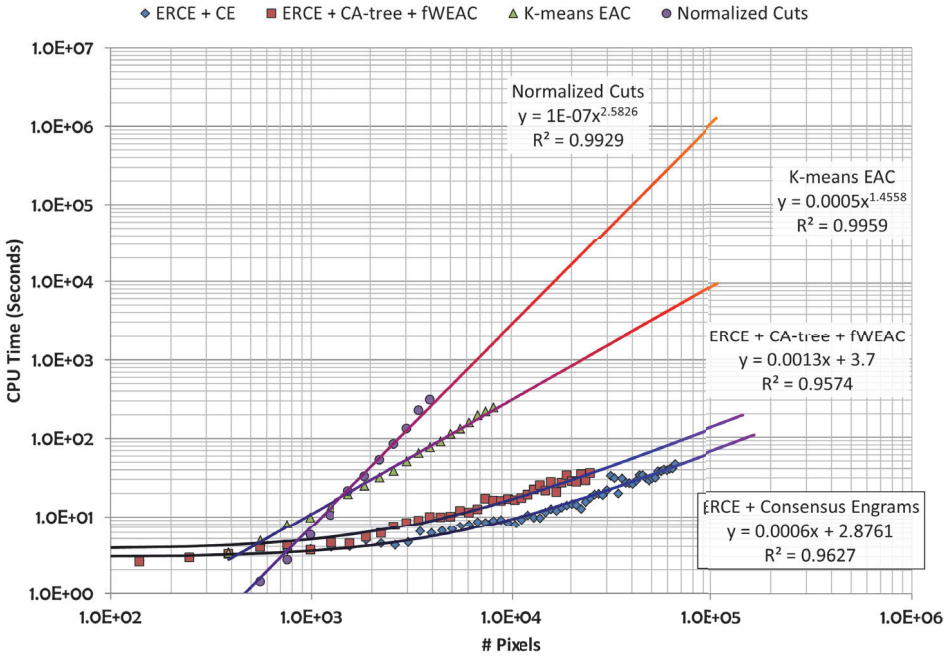


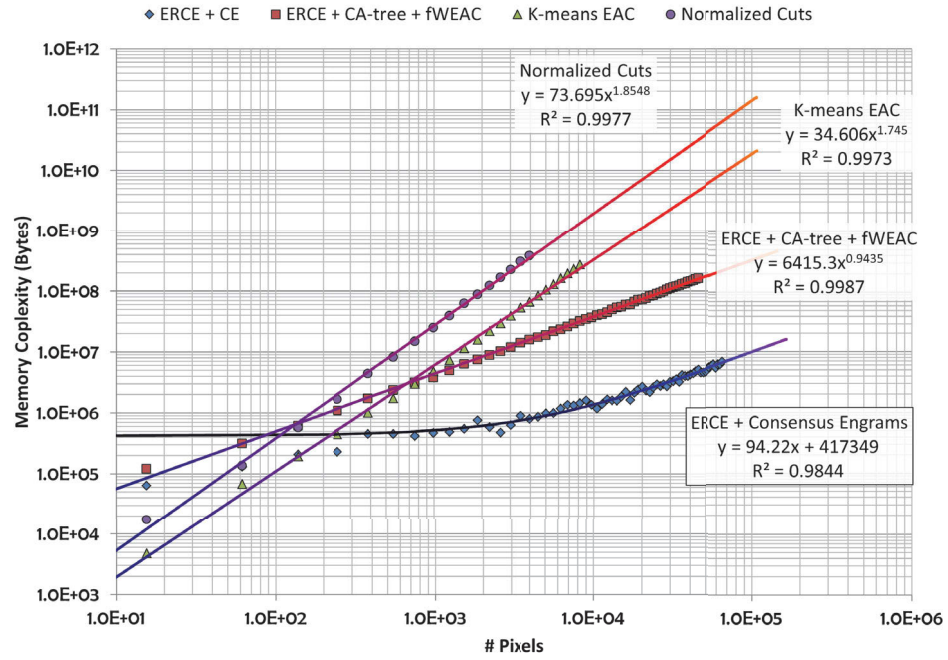
FIGURE 1.11: Memory complexity for clustering 2-dimensional random noise (vol = 1 Byte – 1 Megabytes, d = 2) using various algorithms. The global settings for all algorithms are as follow: the number of representatives (K) = 30; the number of trials/swarms (n_m) = 30.)

The consensus engrams approach is also applied on the blood vessel segmentation from retinal fundus images, using the benchmark dataset from Digital Retinal Images for Vessel Extraction (DRIVE) [80]. The experimental results in Table 6.3 shows that the proposed method is superior in all performance aspects compared to the other methods in the literature, outperforming even Staal's [80] supervised approach to vessel segmentation (ERCE + Consensus Engrams (Unsupervised): 95.58% ± 0.51%; Staal (Supervised): 94.42% ± 0.65%). The ROC curve of the proposed method and other methods in the literature is shown in Figure 1.13. Further information can be seen in Section 6.4.2.

These contributions are summarized in the thesis chapters as shown in the Figure 1.14.



(A) Time Complexity.



(B) Memory Complexity.

FIGURE 1.12: The CPU time and memory required of various algorithms with respect to the number of pixels in an image.

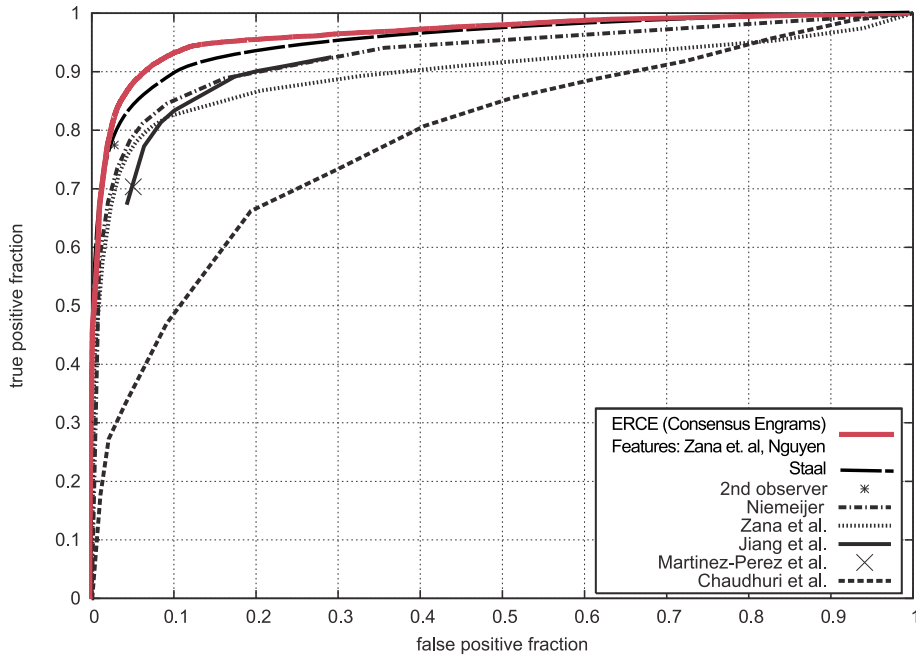


FIGURE 1.13: The ROC curve of the proposed method vs other methods in the literature [80].

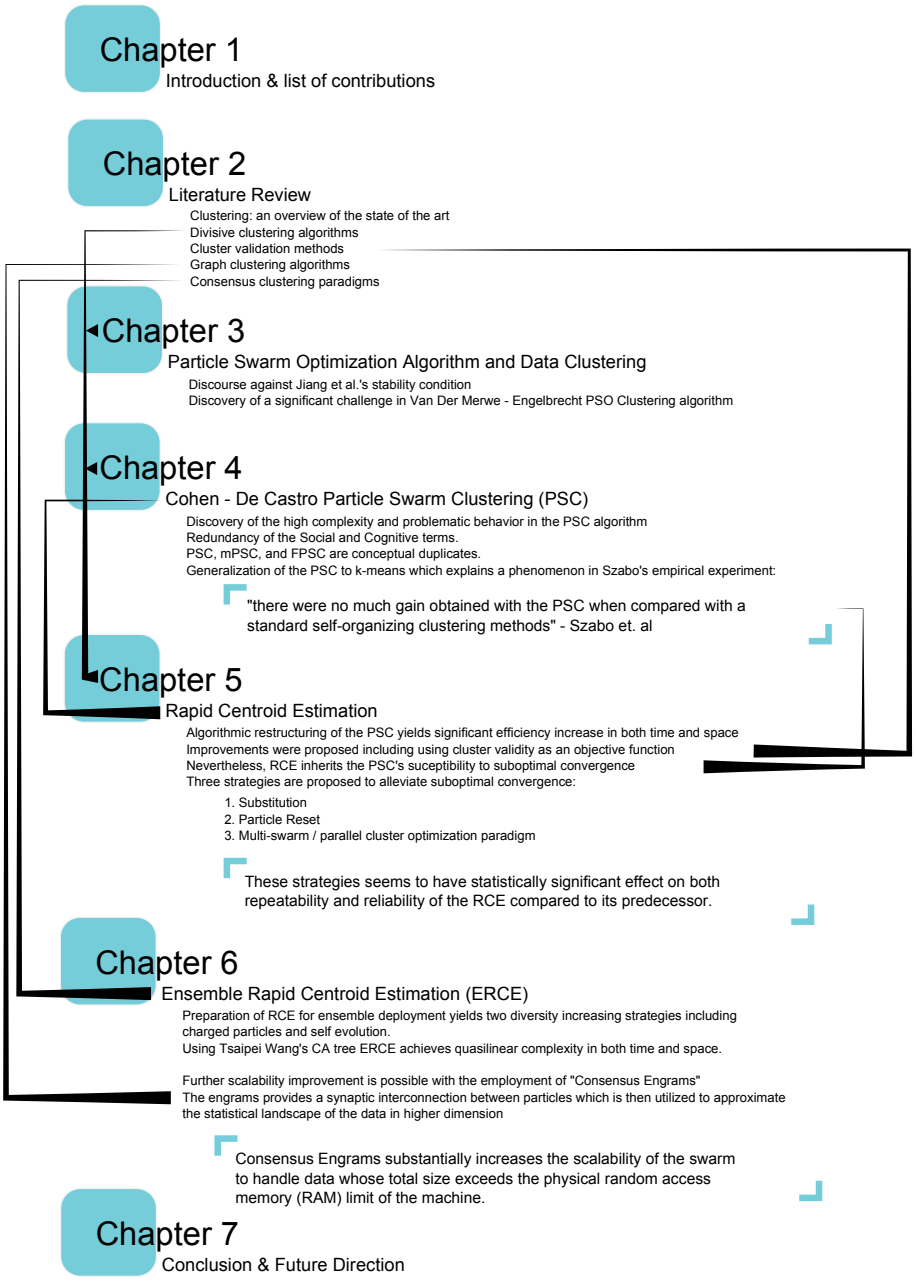


FIGURE 1.14: Research contribution diagram.

Chapter 2

Literature Review

Merriam-Webster defines cluster analysis as "*a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics.*" A *Cluster* is "*a group of things or people that are close together*" [43]. The goal of cluster analysis can be summarized as "*to discover the natural grouping(s) of a set of patterns, points, or objects according to measured or perceived intrinsic characteristics or similarity*" [2]. This chapter is devoted to summarize in great detail the important concepts in clustering algorithms including: Metrics and distance quantization methods (Section 2.2); Divisive clustering algorithms (Section 2.3); Cluster validity indices (Section 2.4); Graph clustering algorithms (Section 2.5); and Consensus clustering paradigms (Section 2.6).

2.1 Challenges in Cluster Analysis

Cluster analysis is exceptionally difficult because of the absence of the definitive ground-truth. Clusters are identified based on a subjective assumption of how *similar* or how *different* data points are. Clusters can differ in terms of their **shape, size, and density**. An illustration on how clusters are defined can be seen in Figure 2.1. In an ideal case, a cluster can be defined as a set of points that is compact and isolated, however this situation is usually not true in real life scenario where the boundary between clusters are often fuzzy/uncertain. Real data contains noises and outliers that pose significant challenge to many clustering algorithms.

A cluster is a subjective entity that is in the eye of the beholder and whose significance and interpretation are dependant on the domain knowledge or predefined expectation [2]. Humans are

adept at conceptual clustering of items and scenarios that are ambiguous or fuzzy. To date there has not been any single clustering algorithm that can outperform humans in visual clustering, scene matching/identification, memory clustering, and sound clustering [2]. However, for things whose information are not sensory perceivable, e.g. gene clustering, we require an extra "tool" that is smart enough to automatically simplify the data to something that is meaningful and perceivable. A desirable clustering algorithm should be able to intuitively cluster objects quickly and effectively subject to our predefined criteria.

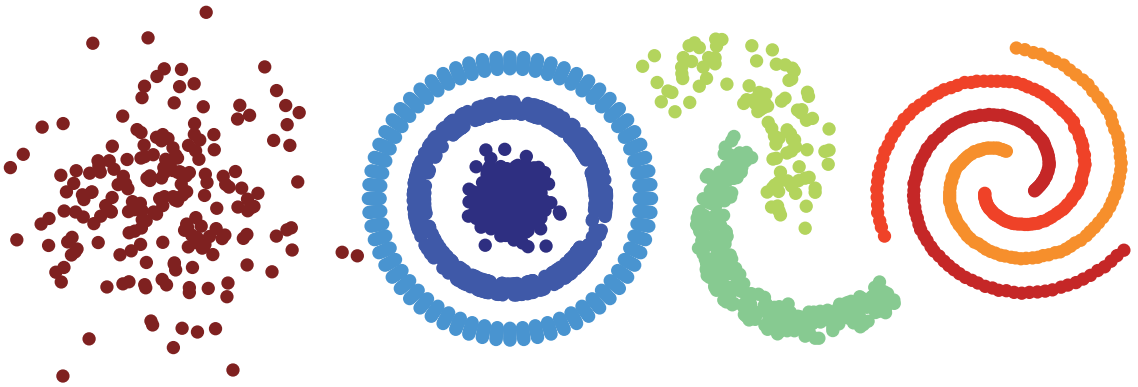


FIGURE 2.1: Clusters differ in term of their shape, size, and density.

Traditional heuristic clustering can be subdivided into two subgroups: agglomerative (bottom-up) clustering and divisive (top-down/prototype based) clustering. Each method offers their own strength and weaknesses. The attempt to increase the efficacy of existing heuristics leads to a new stream of clustering methodology called Consensus/Ensemble Clustering algorithms [64–66, 91, 92]. The ensemble algorithm attempts to combine both divisive and agglomerative approaches to create stable and reliable clustering. The method attempts to recover natural grouping of the cluster based on numerous monte-carlo clustering results from divisive algorithms such as k-means/fuzzy c-means. A number of approaches to cluster optimization have also been proposed by the computational intelligence community including Adaptive Resonance Theory (ART) [93, 94], Self Organizing Map (SOM) [95], Particle Swarm Clustering (PSC) [70, 74, 75], [Ensemble] Rapid Centroid Estimation ([E]RCE) [4, 55, 71–73], and Multi-Objective Clustering with Automatic K-determination (MOCK) [62]. A possible taxonomy of clustering algorithms can be seen in Figure 2.2.

Agglomerative clustering offers an intuitive representation on the relation of one data instance to another in a tree diagram called the *dendrogram*. The drawback of agglomerative clustering is its memory and time complexity which are both quadratic with respect to the volume of the

FIGURE 2.2: A possible taxonomy/interaction map of clustering algorithms

dataset $\mathcal{O}(N^2)$. This quality restricts the application of agglomerative clustering to smaller datasets [3, 55, 66]).

Divisive/Prototype-based clustering algorithms (e.g. k-means, fuzzy c-means) generally have linear complexity $\mathcal{O}(kNt)$ where k and t denote the number of centroids and iterations, respectively [55, 66]. Although the algorithm is non-deterministic, divisive clustering algorithms are attractive because it offers reasonably compact solutions with relatively low computational and memory complexity. However, divisive algorithms do not always reliably cluster the data into its natural clusters because the partitions returned by divisive clusterings varies with multiple initializations

and parameter specifications [3, 64].

Ensemble methods [2, 55, 66, 76] make use of both graph/agglomerative and monte-carlo runs of divisive clusterings. The reports presented in the papers argued that ensemble algorithms are capable for attaining robust solution. The methods allow the estimation of natural grouping and the number of clusters in both convex and non-convex data.

As there are easily hundreds of clustering algorithms available in the literature, for the scope of this thesis, we will focus our discussions only on three niches of unsupervised clustering problems: Graph clustering: including agglomerative hierarchical linkage, graph partitioning, and spectral clustering; Prototype based clustering; and Consensus / Ensemble clustering.

2.2 Quantifying Dissimilarity

Specifying the context on how various data should differ or agree with each other is of critical importance for ensuring optimal clustering. In clustering, the degree of *dissimilarity* between two observation can be described as the *distance* between them. Given a pair of observation vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{dim}$, the pairwise distance $d(\mathbf{x}, \mathbf{y})$ can be quantified using various measures.

2.2.1 Metric Family

Definition 2.2.1 *(Euclidean Distance).* Euclidean distance computes the root of square differences between coordinates of a pair of observations.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}. \tag{2.1}$$

Definition 2.2.2 *(Manhattan distance).* The Manhattan distance [96] defines $d(\mathbf{x}, \mathbf{y})$ as the sum of the absolute differences of their Cartesian coordinates:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{dim} |x_j - y_j|, \tag{2.2}$$

where j denotes the index of dimension, and $|\cdot|$ denotes absolute operator. The Manhattan distance is also known as rectilinear distance, Minkowski's L_1 distance, taxi cab metric, or cityblock distance. Manhattan distance is often used in applications that requires dimension specific distance quantification such as in integrated circuits. The Hamming distance can be considered a Manhattan distance between bit vectors.

Definition 2.2.3 *(Chebyshev distance).* Chebyshev distance is also called Maximum value distance. The distance between two vectors is the greatest difference along any coordinate dimension:

$$d(\mathbf{x}, \mathbf{y}) = \max_{j \in \{1,\ldots,dim\}} |x_j - y_j|, \tag{2.3}$$

where j denotes the index of dimension. The Chebyshev distance is also known as maximum distance, chessboard distance, or Minkowski's L_∞ distance. A comparison between Chebyshev, Manhattan and Euclidean distances can be seen in Figure 2.3.

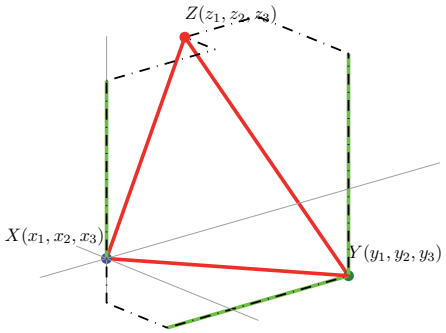


FIGURE 2.3: Pairwise Euclidean distance (red, solid lines), Manhattan distance (black, dot-dashed lines), and Chebyshev distance (green, solid lines).

Definition 2.2.4 *(Minkowski distance).* Minkowski distance is is the generalized metric distance:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{dim} |x_j - y_j|^p \right)^{1/p}, \tag{2.4}$$

j denotes the index of dimension, and $|\cdot|$ denotes absolute operator. The Minkowski distance resembles the normal Euclidean distance formula, except that the p^{th} power and p^{th} root is supplied instead. In fact Minkowski distance is the Euclidean distance when $\mathrm{p} = 2$. It is the Manhattan distance when $\mathrm{p} = 1$, minimum distance when $p \to -\infty$, and maximum (Chebyshev) distance when $p \to +\infty$. Figure 2.4 shows the effect of varying p parameter on the Minkowski distance.

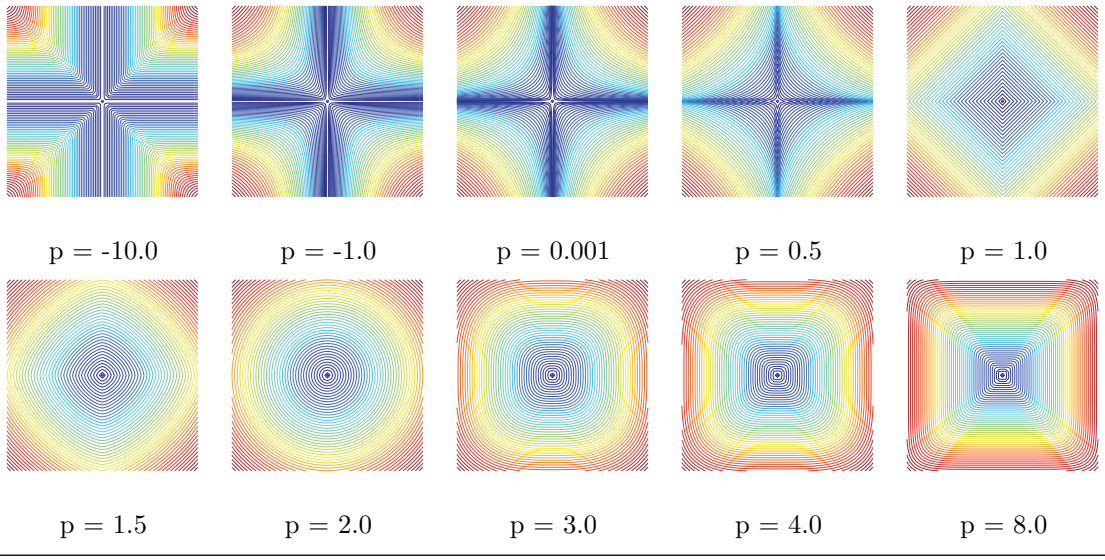


FIGURE 2.4: Minkowski distance with various p.

Definition 2.2.5 *(Mahalanobis distance).* The Mahalanobis distance is a generalization of the normalized Euclidean distance. It measures the normalized distance between the two points in $\mathbb{R}^{dim}$ given the covariance of the cluster. The Mahalanobis distance can be written as follows,

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y})}, \tag{2.5}$$

which generalizes to Euclidean distance if Σ is an identity matrix, and Normalized Euclidean distance if Σ is diagonal. Mahalanobis distance can also be defined as a dissimilarity measure between $\mathbf{x}$ and $\mathbf{y}$ of the same distribution with the covariance matrix Σ.

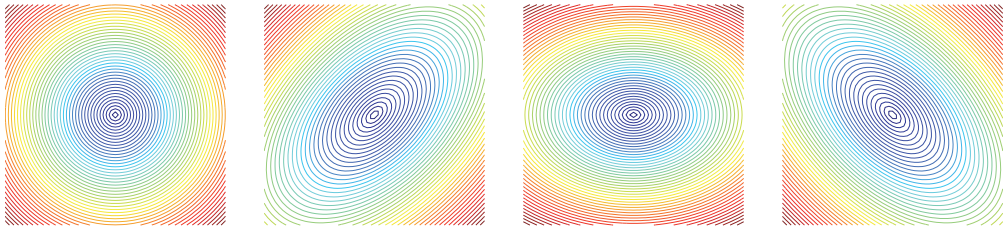


FIGURE 2.5: Mahalanobis distance with various Σ.

2.2.2 Correlation Family

Definition 2.2.6 *(Correlation distance).* The correlation distance is derived from the correlation coefficient ($\rho(\mathbf{x},\mathbf{y})$). The correlation distance ($d(\mathbf{x},\mathbf{y})$) measures the similarity in shape between

the two profiles:

$$d(\mathbf{x}, \mathbf{y}) = 1 - \rho(\mathbf{x}, \mathbf{y}), \tag{2.6}$$

$$\rho_{pearson}(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{\mathbf{xy}}}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}}. \tag{2.7}$$

$$\rho_{spearman}(\mathbf{x}, \mathbf{y}) = 1 - \frac{6\sum_i (r_i(x) - r_i(y))^2}{n(n^2 - 1)}, \tag{2.8}$$

where $r_i(\cdot)$ denotes the rank of the i^{th} dimension.

Correlation measures similarity rather than distance or dissimilarity: It ranges from +1 to -1 where +1 correlation indicates that the two points perfectly match and -1 correlation indicates that the two points perfectly mirror each other. In the context of distance this value is converted to 0 to 2: where 0 indicates +1 correlation, and 2 indicates -1 correlation. Its example usage in binary pattern classification is shown in Figure 2.6.

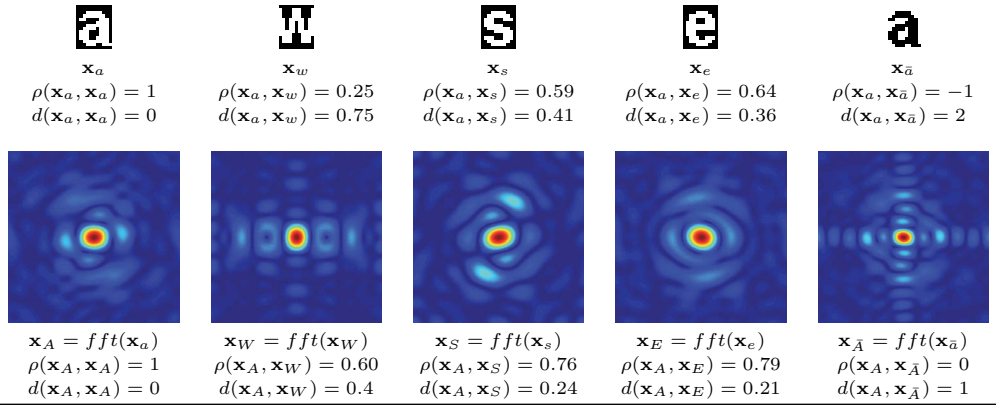


FIGURE 2.6: Correlation distance between binary patterns and their fourier transforms.

Definition 2.2.7 *(Squared Correlation distance)***.** The squared correlation distance is derived similarly from correlation ($\rho(\mathbf{x}, \mathbf{y})$). Contrast to the correlation distance, in the squared correlation distance, both correlated and anti-correlated patterns have the same distance. The greatest separation is achieved when $\mathbf{x}$ and $\mathbf{y}$ are perfectly uncorrelated.

$$d(\mathbf{x}, \mathbf{y}) \quad = \quad 1 - \rho^2(\mathbf{x}, \mathbf{y}), \tag{2.9}$$

Recall the binary image correlation in Figure 2.6. Contrast to correlation distance, the squared correlation distance will designate both a and its perfect negation in the binary space with the distance of 0.

Definition 2.2.8 *(Cosine angle distance).* The cosine angle between the vectors $\mathbf{x}$ and $\mathbf{y}$ is calculated as follows,

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{|\mathbf{x}||\mathbf{y}|}, \tag{2.10}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product, while $|\cdot|$ denotes euclidean norm. The cosine value of 1 indicates that the vector pair is perfectly aligned (0 degrees away from one another), while -1 indicates that the vector pair is 180 degrees away from one another. The quality of cosine similarity thus resembles that of the Pearson correlation. As a distance metric, the cosine angle is converted similarly to correlation as follows

$$d(\mathbf{x}, \mathbf{y}) \quad = \quad 1 - \cos(\mathbf{x}, \mathbf{y}). \tag{2.11}$$

2.2.3 Bregman Divergences

Definition 2.2.9 *(Squared Euclidean Distance).* Squared Euclidean distance computes the squared differences between coordinates of a pair of observations:

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}). \tag{2.12}$$

The squared Euclidean distance puts progressively greater weight on objects that are farther apart. Squared Euclidean distance is a Bregman divergence generated by the convex function $f(\mathbf{x}) = ||\mathbf{x}||^2$. It does not satisfy triangle equality. Its progressive weighting quality is particularly useful for clustering Gaussian type clusters [4, 55, 97]. Squared Euclidean distance is specified as the default metric for the clustering algorithms in MATLAB 2014 [97] and R [98].

Definition 2.2.10 *(Squared Mahalanobis distance).* The Squared Mahalanobis distance is defined as

$$d(\mathbf{x}, \mathbf{y}) = \big(\mathbf{x} - \mathbf{y}\big)^T \Sigma^{-1} \big(\mathbf{x} - \mathbf{y}\big). \tag{2.13}$$

Squared Mahalanobis distance is a Bregman Divergence generated by the function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{Q} \mathbf{x}$, with $\mathbf{Q} = \Sigma^{-1}$. It generalizes to Squared Euclidean distance if Σ is an identity matrix, and Normalized Squared Euclidean distance if Σ is diagonal. Squared Mahalanobis distance can also be defined as a squared dissimilarity measure between $\mathbf{x}$ and $\mathbf{y}$ of the same distribution with the covariance matrix Σ. The Squared Mahalanobis distance does not satisfy triangle equality.

Definition 2.2.11 *(Kullback-Leibler Divergence).* Kullback-Leibler divergence – often abbreviated as KL-divergence – measures the relative entropy between two distributions [88]. Specifically, given two probability distributions $P = p(x)$ and $Q = q(x)$, $p(x) \leq 1, q(x) \leq 1$, KL-divergence $KL(P||Q)$

measures the amount of information lost when Q is used to approximate P. The formulation is as follows,

$$KL(P||Q) = \overbrace{-\sum_x p(x)\log q(x)}^{H(P,Q)} + \overbrace{\sum_x p(x)\log p(x)}^{-H(P)}, \tag{2.14}$$

$$= \sum_x p(x)\log\frac{p(x)}{q(x)}, \tag{2.15}$$

where $H(P,Q)$ denotes the cross entropy between P and Q and $H(P)$ denotes the information entropy of P. KL-divergence is a family of Bregman divergence and it is not appropriately a "distance metric" on the space of probability distributions as it is not symmetric – that is, $KL(P||Q) \neq KL(Q||P)$, – nor does it satisfy the triangle inequality.

The symmetrical version of the KL-divergence was proposed in Kullback's manuscript [88] as follows,

$$\begin{aligned} D_{KL(s)}(P||Q) &= KL(P||Q) + KL(Q||P), \\ &= \sum_x p(x)\log\frac{p(x)}{q(x)} + q(x)\log\frac{q(x)}{p(x)}, \\ &= \sum_x p(x)\log\frac{p(x)}{q(x)} - q(x)\log\frac{p(x)}{q(x)}, \end{aligned} \tag{2.16}$$

Symmetrical KL-divergence is often used in clustering and classification due to its nice symmetry.

Lemma 2.2.1 *(KL-Divergence between multivariate Gaussian distributions)***.** *For continuous d-dimensional multivariate Gaussian distributions* $P(\mathbf{x}) = \mathcal{N}(\mathbf{x};\mu_1,\Sigma_1)$ *and* $Q(\mathbf{x}) = \mathcal{N}(\mathbf{x};\mu_2,\Sigma_2)$*, the KL-divergence between the two is*

$$KL(P||Q) = \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right]. \tag{2.17}$$

Proof: A d-dimensional multivariate Gaussian distribution $\mathcal{N}(\mathbf{x};\mu,\Sigma)$ is defined as follows,

$$\mathcal{N}(\mathbf{x};\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\exp\left(-\frac{(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)}{2}\right), \tag{2.18}$$

where μ and Σ denote its corresponding mean and covariance matrix, respectively.

The KL-divergence between $P(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_1, \Sigma_1)$ and $Q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_2, \Sigma_2)$ can be derived [99],

$$
\begin{aligned}
KL(P||Q) &= \int P(\mathbf{x}) \Big[\log P(\mathbf{x}) - \log Q(\mathbf{x}) \Big] d\mathbf{x} \\
&= E_P \Big[\log P(\mathbf{x}) - \log Q(\mathbf{x}) \Big] \\
&= E_P \left[\log \left(\frac{1}{\sqrt{(2\pi)^d |\Sigma_1|}} \exp \left(-\frac{(\mathbf{x}-\mu_1)^T \Sigma_1^{-1} (\mathbf{x}-\mu_1)}{2} \right) \right) - \dots \right. \\
&\qquad \left. \log \left(\frac{1}{\sqrt{(2\pi)^d |\Sigma_2|}} \exp \left(-\frac{(\mathbf{x}-\mu_2)^T \Sigma_2^{-1} (\mathbf{x}-\mu_2)}{2} \right) \right) \right] \\
&= \frac{1}{2} E_P \Bigg[- \log(2\pi)^d - \log |\Sigma_1| - (\mathbf{x}-\mu_1)^T \Sigma_1^{-1} (\mathbf{x}-\mu_1) + \dots \\
&\qquad \log(2\pi)^d + \log |\Sigma_2| + (\mathbf{x}-\mu_2)^T \Sigma_2^{-1} (\mathbf{x}-\mu_2) \Bigg] \\
&= \frac{1}{2} E_P \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - (\mathbf{x}-\mu_1)^T \Sigma_1^{-1} (\mathbf{x}-\mu_1) + (\mathbf{x}-\mu_2)^T \Sigma_2^{-1} (\mathbf{x}-\mu_2) \right] \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} + E_P \left[-tr \left(\Sigma_1^{-1} (\mathbf{x}-\mu_1)(\mathbf{x}-\mu_1)^T \right) + tr \left(\Sigma_2^{-1} (\mathbf{x}-\mu_2)(\mathbf{x}-\mu_2)^T \right) \right] \right] \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} + E_P \left[-tr \left(\Sigma_1^{-1} \Sigma_1 \right) \right] + E_P \left[tr \left(\Sigma_2^{-1} (\mathbf{x}\mathbf{x}^T - 2\mathbf{x}\mu_2^T + \mu_2 \mu_2^T) \right) \right] \right] \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + tr \left(\Sigma_2^{-1} (\Sigma_1 + \mu_1 \mu_1^T - 2\mu_1 \mu_2^T + \mu_2 \mu_2^T) \right) \right] \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + tr \left(\Sigma_2^{-1} \Sigma_1 \right) - tr (\Sigma_2^{-1} (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T) \right] \\
&= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + tr \left(\Sigma_2^{-1} \Sigma_1 \right) - (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right] \qquad \blacksquare
\end{aligned}
\tag{2.19}
$$

Definition 2.2.12 *(Jensen-Shannon Divergence)*. Jensen–Shannon divergence (JS-divergence), also known as *information radius* [100] or *total divergence to the average* [101], is a Bregman divergence based on the symmetrical Kullback–Leibler divergence with an additional smoothing parameter [102].

The Jensen–Shannon divergence is bounded by 1, given that one uses the base 2 logarithm. The square root of the Jensen–Shannon divergence is a metric often referred to as Jensen-Shannon distance. The formulation is as follows,

$$
D_{JSD}(P||Q) = \frac{KL(P||M) + KL(Q||M)}{2}, \tag{2.20}
$$

where M denotes a mixture distribution of Q and P,

$$
M = \frac{1}{2}(P + Q). \tag{2.21}
$$

2.2.4 Distance Between Logical/Categorical Vectors

Definition 2.2.13 *(Hamming Distance).* The Hamming distance [103] between two binary vectors $\mathbf{x}$ and $\mathbf{y}$ is the number of bits/categories that differs,

$$d_{hamming}(\mathbf{x}, \mathbf{y}) = |\mathbf{x} \neq \mathbf{y}|. \tag{2.22}$$

Definition 2.2.14 *(Normalized Hamming Distance).* The normalized Hamming distance [103] divides $d_{hamming}(\mathbf{x}, \mathbf{y})$ with the number of bits/categories such that it is normalized between $[0, 1]$ where 0 indicates perfect synchrony. The formulation is as follows,

$$d_{norm\,hamming}(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \neq \mathbf{y}|}{|\mathbf{x}|}. \tag{2.23}$$

Definition 2.2.15 *(Jaccard Distance).* The Jaccard index or the Jaccard similarity index/coefficient measures similarity between finite sample sets, and is defined as the size of the intersection (non-zero categories that are equal) divided by the size of the union of the sample sets (number of non-zero categories). The Jaccard distance is calculated simply as 1 - Jaccard similarity. The formulation is as follows,

$$J(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \bigcap \mathbf{y}|}{|\mathbf{x} \bigcup \mathbf{y}|} \tag{2.24}$$

$$d_{Jaccard} = 1 - J(\mathbf{x}, \mathbf{y}). \tag{2.25}$$

Definition 2.2.16 *(Sørensen-Dice Distance).* The Sørensen-Dice index [104, 105] closely resembles the Jaccard similarity index/coefficient. The Sørensen-Dice distance is calculated simply as 1 - Sørensen-Dice similarity. The formulation is as follows,

$$s(\mathbf{x}, \mathbf{y}) = \frac{2|\mathbf{x} \bigcap \mathbf{y}|}{|\mathbf{x}| + |\mathbf{y}|} \tag{2.26}$$

$$d_{\text{Sørensen-Dice}} = 1 - s(\mathbf{x}, \mathbf{y}). \tag{2.27}$$

2.2.5 Mutual Information

Definition 2.2.17 *(Mutual Information).* Mutual information between two discrete probability distribution $I(X;Y)$ decodes the statistical information shared between two data vectors: $\mathbf{x}$ generated by the distribution X; and $\mathbf{y}$ generated by the distribution Y [106]. This measure is a useful

indication of coherence between the two distributions that generates the vectors. The mutual information is often used to compare clustering results in a cluster ensemble setting [107–109]. An informative explanatory illustration can be seen in Figure 2.7.

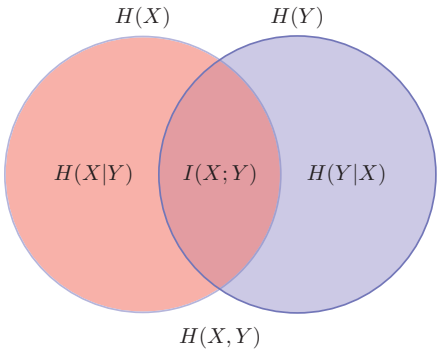


FIGURE 2.7: Mutual Information $I(X;Y)$ between X and Y.

As described by Shannon, the information entropy of a random variable is a function which characterizes the "unpredictability" of the probability distribution [89],

$$H(X) = \sum_{x \in X} p(x) \log p(x). \tag{2.28}$$

The cross entropy is computed similarly using the joint probability distribution,

$$H(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(x,y). \tag{2.29}$$

Mutual information applies Shannon's information entropy for examining the dependence between X and Y. In Figure 2.7 maximizing mutual information is equal to minimizing the KL-divergence between the cross-entropy $H(X,Y)$ and the marginal entropies: $H(X)$ and $H(Y)$ where $p(x) = \sum_{y \in Y} p(x,y)$ and $p(y) = \sum_{x \in X} p(x,y)$, obtained from marginalization of the joint probability. The

formulation is as follows,

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y), \\
&= H(Y) - H(Y|X), \\
&= H(X) + H(Y) - H(X,Y), \\
&= H(X,Y) - H(X|Y) - H(Y|X), \\
&= \overbrace{-\sum_{x\in X} p(x)\log p(x)}^{H(X)} \overbrace{-\sum_{y\in Y} p(y)\log p(y)}^{H(Y)} + \overbrace{\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x,y)}^{-H(X,Y)}, \\
&= -\sum_{x\in X} \underbrace{\sum_{y\in Y} p(x,y)}_{p(x)} \log p(x) - \sum_{y\in Y} \underbrace{\sum_{x\in X} p(x,y)}_{p(y)} \log p(y) + \sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x,y), \\
&= \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}, \\
&= KL(p(x,y)||p(x)p(y)).
\end{aligned}
\tag{2.30}
$$

Definition 2.2.18 *(Normalized Mutual Information).* The normalized mutual information [107] attempts to scale the mutual information such that 0 indicates independence and 1 indicates perfect dependence. There are varying formulations for normalized mutual information [107–110] as follows,

$$NMI_{\text{Strehl-Ghosh}}(X,Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}} \tag{2.31}$$

$$NMI_{\text{Kvalseth 1}}(X,Y) = \frac{2I(X;Y)}{H(X)+H(Y)}, \tag{2.32}$$

$$NMI_{\text{Kvalseth 2}}(X,Y) = \frac{I(X;Y)}{\min(H(X),H(Y))}, \tag{2.33}$$

$$NMI_{\text{Kvalseth 3}}(X,Y) = \frac{I(X;Y)}{\max(H(X),H(Y))}, \tag{2.34}$$

$$NMI_{\text{Yao}}(X,Y) = \frac{I(X;Y)}{H(X,Y)}, \tag{2.35}$$

which obviously resides inside the bounds $0 \leq NMI \leq 1$. Figure 2.8 illustrates how the normalized mutual information values the mutual information between the sets.

Definition 2.2.19 *(Variation of Information).* Variation of information (VoI) [108, 109] converts the mutual information into a metric, satisfying triangle equality, non-negativity, indiscernibility and symmetry. Variation of information is suitable when a cluster is defined by the coherence between the probability distributions. The (unnormalized) variation of information $d_{VoI(u)}$ is

defined as follows,

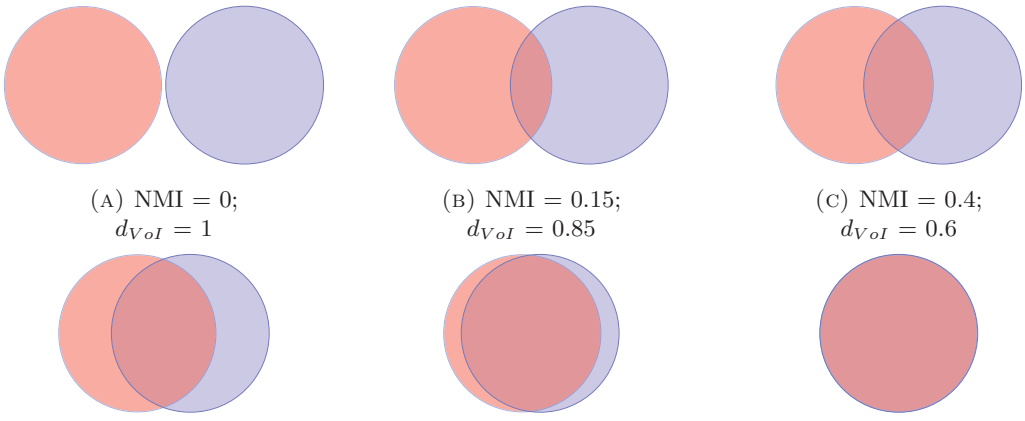
$$\begin{aligned} d_{VoI(u)}(X,Y) &= H(X,Y) - I(X;Y), \\ &= H(X) + H(Y) - 2I(X;Y), \\ &= H(X|Y) + H(Y|X). \end{aligned} \tag{2.36}$$

Realizing $d_{VoI} \leq H(X,Y)$, the normalized formulation can be formulated as follows,

$$d_{VoI}(X,Y) = 1 - \frac{I(X;Y)}{H(X,Y)}, \tag{2.37}$$

$$= 1 - NMI(X,Y), \tag{2.38}$$

which bounds d_{VoI} from 0 to 1.



(A) NMI = 0; $d_{VoI} = 1$ (B) NMI = 0.15; $d_{VoI} = 0.85$ (C) NMI = 0.4; $d_{VoI} = 0.6$

(D) NMI = 0.75; $d_{VoI} = 0.25$ (E) NMI = 0.95; $d_{VoI} = 0.05$ (F) NMI = 1; $d_{VoI} = 0$

FIGURE 2.8: Conceptual illustration of the normalized mutual information.

2.3 Divisive Clustering

Divisive clustering starts by assuming the whole dataset as one monolithic cluster, which is then divided into pieces/partitions. In each of these partitions, an encoding prototype is assigned. Collectively, the prototypes impose voronoi tessellation on the data, were each voronoi cell encodes the contained objects inside the corresponding voronoi region, compressing the information in the particular region using a compact representative called a *centroid*.

Definition 2.3.1 *(Voronoi Regions).* The representatives $\mathbb{C}$ induces a *voronoi partition* of $\mathbb{R}^{dim}$, which is a decomposition of $\mathbb{R}^{dim}$ into k convex *cells* [111]. Each $\mathbb{C}_z$ is governed by the corresponding center $\mathbf{z} \in \mathbb{C}$ and contains the region of space whose nearest representative is $\mathbf{z}$.

$\mathbb{C}$ induces optimal clustering on the dataset, $\mathbb{S} = \bigcup_{\mathbf{z}\in\mathbb{C}} \mathbb{C}_z$, where

$$\mathbb{C}_z = \{\mathbf{x} \in \mathbb{S} : \text{ the closest representative of } \mathbf{x} \text{ is } \mathbf{z}\}. \tag{2.39}$$

The objective function of a divisive algorithm can then be quantified as a function of distance from each $\mathbf{x} \in \mathbb{C}_z$ to its corresponding cell center $\mathbf{z}$,

$$f(\mathbb{C}) = \sum_{\mathbf{z}\in\mathbb{C}} \sum_{\mathbf{x}\in\mathbb{C}_z} d(\mathbf{x}, \mathbf{z}), \tag{2.40}$$

For K partitions of $\mathbb{S}$ the objective function is for example,

$$f(\mathbb{C}, K) = f(\mathbb{C}_1, \ldots, \mathbb{C}_z; \mathbf{z}_1, \ldots, \mathbf{z}_K) = \sum_{j=1}^{K} \sum_{\mathbf{x}\in\mathbb{C}_j} d(\mathbf{x}, \mathbf{z}_j), \tag{2.41}$$

which would be minimized when,

$$\forall j : \mathbf{z}_j = mean(\mathbb{C}_j) = \sum_{\mathbf{x}\in\mathbb{C}_j} \frac{\mathbf{x}}{|\mathbb{C}_j|}. \tag{2.42}$$

2.3.1 k-means

k-means [112] is arguably the simplest of all divisive prototype-based clustering algorithm. The main idea is to partition the data into voronoi regions using K centroids, one for each cell. k-means aims at minimizing the within-cluster sum of squares,

$$\begin{aligned} f(\mathbb{C}, K) &= \sum_{j=1}^{K} \sum_{i=1}^{N} \mathsf{u}_{ij} ||\mathbf{x}_i - \mathbf{z}_j||^2, \\ &= \sum_{j=1}^{K} \sum_{i=1}^{N} \mathsf{u}_{ij} d_{ij} \qquad\qquad \mathsf{u}_{ij} \in [0, 1], \end{aligned} \tag{2.43}$$

where $\mathbf{z}_j : \forall j \in \{1, \ldots, K\}$ denotes the cluster centers, $\mathbf{x}_i : \forall i \in \{1, \ldots, N\}$ denotes the observations, d_{ij} denotes the distance between $\mathbf{x}_i$ and $\mathbf{z}_j$, where, in case of squared Euclidean distance, can be written as $||\mathbf{x}_i - \mathbf{z}_j||^2$. The matrix $\mathsf{u}_{ij} \in [0, 1]$ is a $K \times N$ binary membership matrix where $\mathsf{u}_{ij} = 1$ denotes that $\mathbf{x}_i \in \mathbb{C}_j$, whereas $\mathsf{u}_{ij} = 0$ denotes the otherwise, $\mathbf{x}_i \notin \mathbb{C}_j$. In k-means, each voronoi region is crisply partitioned, which means each observation can only belong to a cell with the *nearest* center such that the condition $\sum_{j=1}^{K} \mathsf{u}_{ij} = 1 : \forall i$ holds at all times. In the case where equidistant points exist (e.g. given i and j, $\left|\{\forall k : d_{ik} = d_{ij}\}\right| > 1$), selection can be made on

any point because such state is equivalent to *don't care* ($\times$) in logic circuits, e.g. $\mathsf{u}_{ij} \leftarrow \times$. In practice, the assignment would depend on the programmatical implementation. However, it needs to be noted that the effect of this dilemma to the final output is trivial.

The k-means algorithm can be considered as a family of the expectation maximization algorithm. The first step, called the **assignment (expectation) step**, updates the U matrix by assigning each $\mathbf{x}$ to the nearest $\mathbf{z}$; The following step, called the **update (maximization) step**, reassigns each centroids to the mean of the corresponding cluster. The process is repeated until local optimum is reached, or when $\partial f(\mathbb{C}, K)/\partial t = 0$. The pseudocode of the k-means algorithm can be seen in Algorithm 2.1.

Algorithm 2.1 k-means pseudocode

Input: Data matrix $\mathbf{X}$
Output: Centroid matrix $\mathbf{Z}$ and membership matrix U.
1: Initialize centers $\mathbf{Z} \in \mathbb{R}^{dim}$ at random within the search space,
2: **repeat**
3: Given $D(\mathbf{X}, \mathbf{Z})$, update U such that $\forall j : \mathbb{C}_j \leftarrow \{\mathbf{x} \in \mathbb{S}$ whose closest center is $\mathbf{z}_j\}$
4: $\forall j : \mathbf{z}_j \leftarrow mean(\mathbb{C}_j)$
5: **until** $\partial f(\mathbb{C}, K)/\partial t = 0$

In general case, k-means returns a locally optimum partition $\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_K\}$ which satisfies Equation 2.42 but might not necessarily be the desired encoding for $\mathbb{S}$.

2.3.1.1 Proof of Convergence

The proof below summarizes [113].

Lemma 2.3.1 *(Convergence of k-means). Given an initial position $\mathbf{z}_{1,\ldots,K}$, the k-means algorithm is guaranteed to converge to*

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} \mathsf{u}_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} \mathsf{u}_{ij}} = mean(\mathbb{C}_j), \tag{2.44}$$

which minimizes the average distortion within the voronoi region.

Proof: The convergence of k-means can be shown by proving that the distortion $f(\mathbb{C}, K)$ monotonically decreases with each iteration. k-means convergence is guaranteed if $\exists \mathbf{Z} : \dfrac{\partial f(\mathbb{C}, K)}{\partial \mathbf{Z}} = 0$

for each update and reassignment. The proof of this lemma is as follows,

$$f(\mathbb{C}, K) = \sum_{j=1}^{K} \sum_{i=1}^{N} \mathsf{u}_{ij} ||\mathbf{x}_i - \mathbf{z}_j||^2, \tag{2.45}$$

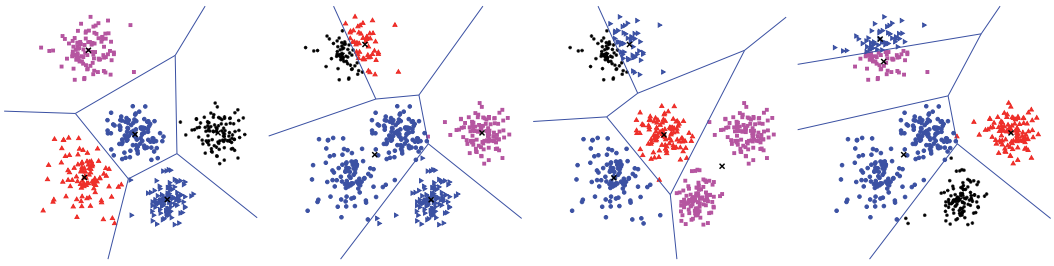
$$\frac{\partial f(\mathbb{C}, K)}{\partial \mathbf{Z}} = \sum_{j=1}^{K} \sum_{i=1}^{N} 2\mathsf{u}_{ij}(\mathbf{x}_i - \mathbf{z}_j) = 0, \tag{2.46}$$

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} \mathsf{u}_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} \mathsf{u}_{ij}} = mean(\mathbb{C}_j), \tag{2.47}$$

which shows that the k-means update step will always converge to the local minima. At the reassignment step, each $\mathbf{x}$ is assigned to the nearest $\mathbf{z}$ which will also decrease $f(\mathbb{C}, K)$. Since there is only a finite set of possible clusterings, $f(\mathbb{C}, K)$ monotonically decreases until eventually arrive at a local minimum. ■

2.3.2 k-means++

k-means algorithm is very sensitive to its initialization parameters [114]. There are scenarios where the k-means algorithm failed to converge to the optimal cluster as can be seen in Figure 2.9.



(A) Desired (B) Suboptimal-1 (C) Suboptimal-2 (D) Suboptimal-3

FIGURE 2.9: Suboptimal partitions returned by k-means due to poor initialization

Arthur and Vassilvitskii proposed the k-means++ initialization method [114] which can be explained as follows: Pick the K centers one at a time, afterwards choose each point at random, with probability proportional to its squared distance from the already chosen centers. Arthur and Vassilvitskii reported that using this seeding technique, significant improvements can be made to the original algorithm [114]. The complete pseudocode of the algorithm can be seen in Algorithm 2.2.

Algorithm 2.2 k-means++ pseudocode

Input: $\mathbf{X}$, # prototypes K
Output: Centroid matrix $\mathbf{Z}$ and membership matrix U.

pick a point $\mathbf{x} \in \mathbb{S}$ uniformly at random and set $\mathbb{C}_z \leftarrow \{\mathbf{x}\}$ // k-means++ initialization
while $|\mathbb{C}| < k$
  pick another $\mathbf{x} \in \mathbb{S}$ at random, with probability proportional to $f(\mathbb{C}; \{\mathbf{x}\}) = \min_{\mathbf{z} \in \mathbb{C}} ||\mathbf{x} - \mathbf{z}||^2$.
  $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathbf{x}\}$
end while // regular k-means routine
repeat
  Given $D(\mathbf{X}, \mathbf{Z})$, update U such that $\forall j : \mathbb{C}_j \leftarrow \{\mathbf{x} \in \mathbb{S}$ whose closest center is $\mathbf{z}_j\}$
  $\forall j : \mathbf{z}_j \leftarrow mean(\mathbb{C}_j)$
until $\partial f(\mathbb{C}, K)/\partial t = 0$

2.3.3 Fuzzy c-means

Fuzzy c-means is the fuzzy counterpart of the k-means algorithm [115]. While in k-means an observation can only belong to one distinct voronoi region, in fuzzy c-means it can belong to multiple voronoi regions with a specific degree of membership that is inversely proportional to its relative distance to the centroid vectors.

The general formulation of the fuzzy c-means clustering is as follows,

$$f(\mathbb{C}, C, m) = \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^{m} ||\mathbf{x}_i - \mathbf{z}_j||^2 \qquad u_{ij} \in \{0, 1\}, \tag{2.48}$$

$$u_{ij} = \frac{d_{ij}^{-\frac{1}{m-1}}}{\sum_{k=1}^{C} d_{ik}^{-\frac{1}{m-1}}}, \tag{2.49}$$

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^{m}}. \tag{2.50}$$

where $\mathbf{z}_j : \forall j \in \{1, \ldots, C\}$ denotes the cluster centers, $\mathbf{x}_i : \forall i \in \{1, \ldots, N\}$ denotes the observations, d_{ij} denotes the distance between $\mathbf{x}_i$ and $\mathbf{z}_j$, where, in case of squared Euclidean distance, can be written as $||\mathbf{x}_i - \mathbf{z}_j||^2$. m is a constant which controls the degree of fuzziness, higher m would result in fuzzier partitions. The matrix U is a $C \times N$ membership matrix, $u_{ij} \in \{0, 1\}, u_{ij} \in \mathbb{R}$, where each element u_{ij} denotes the degree of membership of the point $\mathbf{x}_i$ relative to the cluster $\mathbb{C}_j$. In fuzzy c-means, an observation can belong to multiple cells, each with a specific degree of membership, where $u_{ij} \to 1$ denotes the highest degree of membership and $u_{ij} \to 0$ the lowest degree of membership. The constraints $u_{ij} \in \{0, 1\}$ and $\forall i : \sum_{j=1}^{C} u_{ij} = 1$ must hold at all times.

Optimizing Equation 2.48 can be done similarly to k-means using **Expectation Maximization (EM)** based on Equation 2.49 and Equation 2.50 until the error gradient is lower than a given threshold ε, such that $-\partial f(\mathbb{C}, C, m)/\partial t < \varepsilon$. The pseudocode for the fuzzy c-means can be seen in Algorithm 2.3. Fuzzy c-means clustering results with varying m on the five Gaussians dataset is shown in Figure 2.10.

Algorithm 2.3 Fuzzy c-means pseudocode

Input: $\mathbf{X}$, # prototypes C, fuzzifier constant m, max. iteration t_{max}, and gradient threshold ε.
Output: Centroid matrix $\mathbf{Z}$ and membership matrix U.

1: Initialize centers $\mathbf{Z} \in \mathbb{R}^{dim}$ at random within the search space,
2: **repeat**
3: Given $\mathbf{D} = D(\mathbf{X}, \mathbf{Z})$, update U such that $\forall i, j : u_{ij} \leftarrow \dfrac{d_{ij}^{-\frac{1}{m-1}}}{\sum_{k=1}^{C} d_{ik}^{-\frac{1}{m-1}}}$,
4: $\forall j : \mathbf{z}_j \leftarrow \dfrac{\sum_{i=1}^{N} u_{ij}^{m} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^{m}}$.
5: **until** $(-\partial f(\mathbb{C}, C, m)/\partial t < \varepsilon) \vee (t > t_{max})$

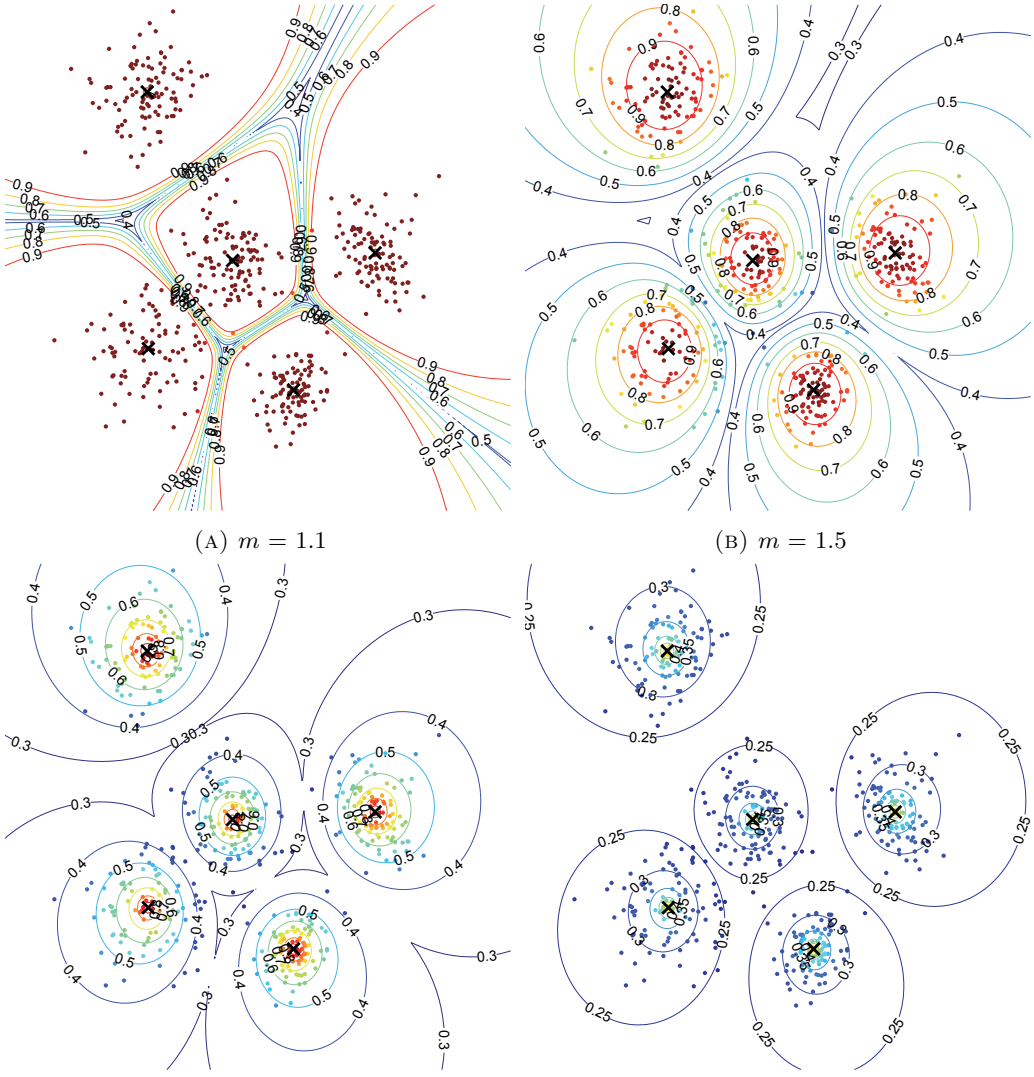
2.3.3.1 Proof of convergence

Fuzzy c-means convergence is guaranteed when $\exists\{U, \mathbf{Z}\}$ that minimizes Equation 2.48 on each Expectation and Maximization step.

Derivations in this subsection are personally done by the author using the method outlined in Bishop's Pattern Recognition and Machine Learning [116]. The book was used as an important reference [116]. The proofs are as follows.

Expectation Step

Lemma 2.3.2 *(Optimality of the Expectation Step).* *Given a fix* $D(\mathbf{X}, \mathbf{Z})$, $f(U, D(\mathbf{X}, \mathbf{Z}), m)$ *is optimized when* $\forall i, j; \quad u_{ij} = \dfrac{d_{ij}^{-\frac{1}{m-1}}}{\sum_{k=1}^{C} d_{ik}^{-\frac{1}{m-1}}}$.



(A) $m = 1.1$ (B) $m = 1.5$

(C) $m = 2$ (D) $m = 4$

FIGURE 2.10: Fuzzy c-means result on the five Gaussians dataset (C=5)

Proof: Given $D(\mathbf{X}, \mathbf{Z})$, we express Equation 2.48 as a constrained optimization problem,

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & f(U, D(\mathbf{X}, \mathbf{Z}), m) = \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^{m} d_{ij}, \\ \text{s.t.} \quad & u_{ij} \in \{0, 1\}, \\ \text{s.t.} \quad & \sum_{j=1}^{C} u_{ij} = 1, \end{aligned} \tag{2.51}$$

which can be simplified by introducing the Lagrange multiplier $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$,

$$L(U, D(\mathbf{X}, \mathbf{Z}), m, \Lambda) = \sum_{j=1}^{C}\sum_{i=1}^{N} u_{ij}^{m} d_{ij} + \sum_{i=1}^{N} \lambda_i \left(1 - \sum_{j=1}^{C} u_{ij}\right), \tag{2.52}$$

where the Karush-Kuhn-Tucker (KKT) necessary condition for the minimum is that the partial derivative of the Lagrange function $L(U, D(\mathbf{X}, \mathbf{Z}), m, \Lambda)$ (Equation 2.52) w.r.t u_{ij} vanishes at u_{lk} such that,

$$\frac{\partial}{\partial u_{ij}} L(U, D(\mathbf{X}, \mathbf{Z}), m, \Lambda) = m u_{lk}^{m-1} d_{lk} - \lambda_l \doteq 0. \tag{2.53}$$

It then follows that

$$u_{lk} = \left(\frac{\lambda_l}{m d_{lk}}\right)^{\frac{1}{m-1}}. \tag{2.54}$$

Imposing the constraint $\forall i : \sum_{j=1}^{C} u_{ij} = 1$ to Equation 2.54 gives

$$\begin{aligned} \sum_{k=1}^{C} \left(\frac{\lambda_l}{m d_{lk}}\right)^{\frac{1}{m-1}} &= 1, \\ \lambda_l &= \left(\sum_{k=1}^{C} (m d_{lk})^{\frac{1}{m-1}}\right)^{m-1}. \end{aligned} \tag{2.55}$$

Finally, given that $l = i$, substituting λ_l, $l \in \{1, \ldots, N\}$ to Equation 2.54 proves Lemma 2.3.2,

$$u_{ij} = \left(\frac{\left(\sum_{k=1}^{C} (m d_{ik})^{\frac{1}{m-1}}\right)^{m-1}}{m d_{ij}}\right)^{\frac{1}{m-1}} = \frac{d_{ij}^{-\frac{1}{m-1}}}{\sum_{k=1}^{C} d_{ik}^{-\frac{1}{m-1}}}. \quad \blacksquare \tag{2.56}$$

Maximization Step

Lemma 2.3.3 *(Optimality of the Maximization Step). Given a fixed U, $f(U, D(\mathbf{X}, \mathbf{Z}), m)$ is optimized when is optimized when* $\mathbf{z}_j = \dfrac{\sum_{i=1}^{N} u_{ij}^{m} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^{m}}$.

Proof:

Given U, Equation 2.48 is minimized when the gradient w.r.t $\mathbf{z}_j : j \in \{1, \ldots, C\}$ vanish,

$$\begin{aligned}
\nabla_{\mathbf{z}_j} f(U, D(\mathbf{X}, \mathbf{Z}), m) &= \nabla_{\mathbf{z}_j} \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^m d_{ij}, \\
&= \nabla_{\mathbf{z}_j} \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^m (\mathbf{x}_i - \mathbf{z}_j)^T (\mathbf{x}_i - \mathbf{z}_j), \\
&= -2 \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^m (\mathbf{x}_i - \mathbf{z}_j), \\
&= -2 \sum_{j=1}^{C} \left(\sum_{i=1}^{N} u_{ij}^m \mathbf{x}_i - \sum_{i=1}^{N} u_{ij}^m \mathbf{z}_j \right) \qquad \doteq \quad 0.
\end{aligned} \tag{2.57}$$

It then follows that solving $\mathbf{z}_j : j \in \{1, \ldots, C\}$ proves Lemma 2.3.3,

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} u_{ij}^m \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^m}. \quad \blacksquare \tag{2.58}$$

2.3.4 Soft k-means

Soft k-means closely resembles fuzzy c-means. In fact, the only difference between soft k-means and Fuzzy c-means is that soft k-means expresses fuzziness using exponential family [117]. The general formulation for soft k-means is as follows,

$$f(\mathbb{C}, K, \beta) = \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij} ||\mathbf{x}_i - \mathbf{z}_j||^2 \qquad u \in \{0, 1\}, \tag{2.59}$$

$$u_{ij} = \frac{\exp(-\beta d_{ij})}{\sum_{k=1}^{K} \exp(-\beta d_{ik})}, \tag{2.60}$$

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}, \tag{2.61}$$

where $\mathbf{X}, \mathbf{Z}$ are observation and centroid matrix. β is the *stiffness* parameter. $U = \{\cup_{\forall i,j} u_{ij} \in \{0, 1\}\}$ is a responsibility matrix. In soft k-means, distance pairs are modeled using a univariate

Gaussian with equal variance, such that $\forall j, \beta = 1/(2\sigma_j^2)$,

$$\begin{aligned} p(\mathbf{x}_i|\mathbf{z}_j, \sigma_j) &= \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{||\mathbf{x}_i - \mathbf{z}_j||^2}{2\sigma_j^2}\right), \\ &= \frac{1}{\sqrt{\pi/\beta}} \exp\left(-\beta d_{ij}\right) \qquad\qquad \beta = \frac{1}{2\sigma_j^2}. \end{aligned} \tag{2.62}$$

This univariate distribution represents a hypersphere $\in \mathbb{R}^{dim}$, where it is assumed that the representatives are equally distributed in all dimensions. This formula is analogue to a multivariate Gaussian distribution with zero covariances and equal variances,

$$p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) = \frac{1}{\sqrt[dim]{2\pi|\Sigma_j|}} \exp\left(-\frac{(\mathbf{x}_i - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mathbf{z}_j)}{2}\right). \tag{2.63}$$

The result of fuzzy c-means and soft k-means on five Gaussians dataset can be seen in Figure 2.11. The soft k-means pseudocode can be seen in Algorithm 2.4.

Algorithm 2.4 Soft k-means pseudocode

Input: $\mathbf{X}$, # prototypes K, fuzzifier constant β, max. iteration t_{max}, and gradient threshold ε.
Output: Centroid matrix $\mathbf{Z}$ and responsibility matrix R.

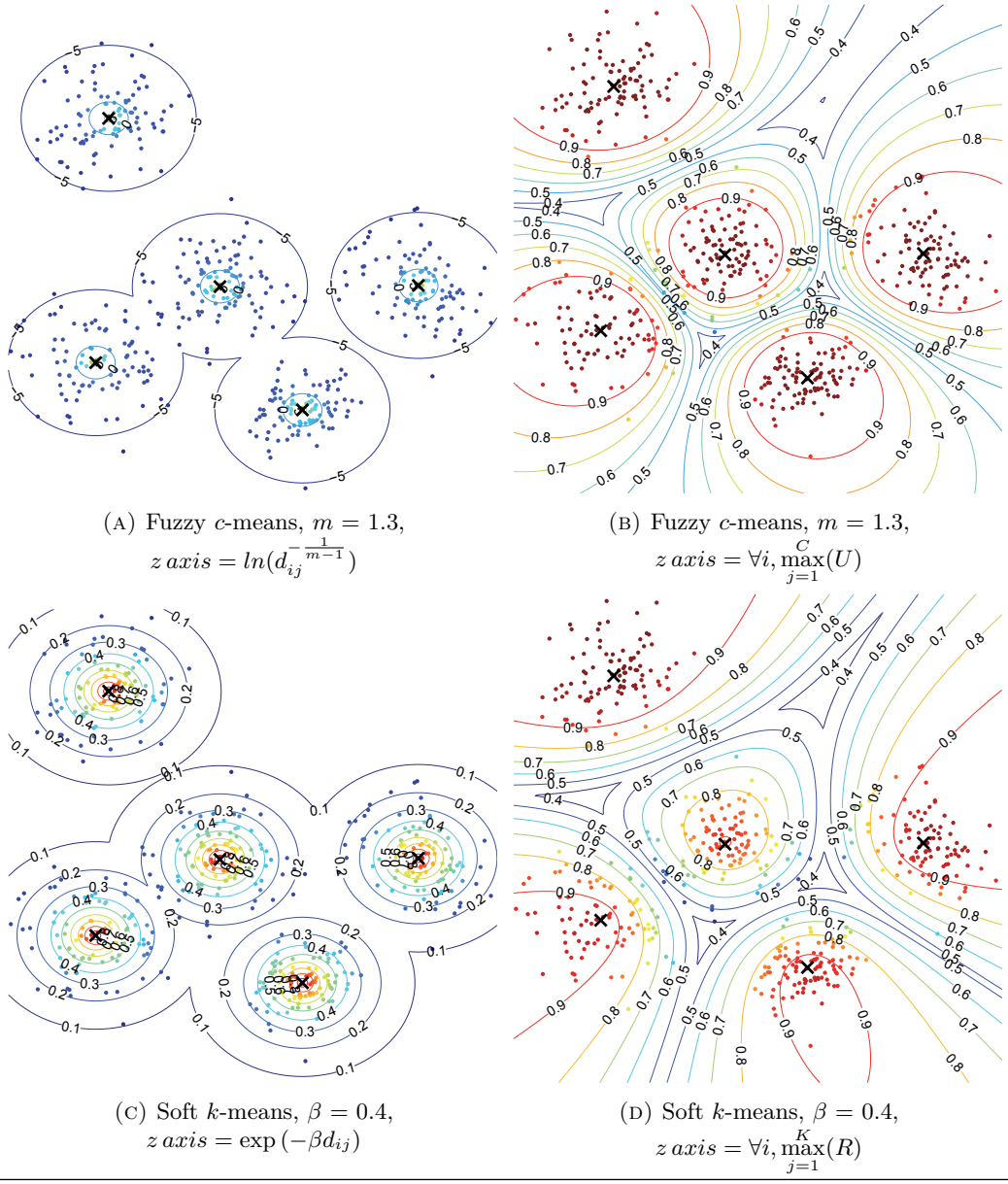
1: Initialize centers $\mathbf{Z} \in \mathbb{R}^{dim}$ at random within the search space,
2: **repeat**
3: Given $\mathbf{D} = D(\mathbf{X}, \mathbf{Z})$, update U such that $\forall i, j : u_{ij} \leftarrow \frac{\exp(-\beta d_{ij})}{\sum_{k=1}^{K} \exp(-\beta d_{ik})}$,
4: $\forall j : \mathbf{z}_j \leftarrow \frac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}$.
5: **until** $(-\partial f(\mathbb{C}, K, \beta)/\partial t < \varepsilon) \vee (t > t_{max})$

2.3.4.1 Proof of Convergence

Derivations in this subsection are personally done by the author using the method outlined in Bishop's Pattern Recognition and Machine Learning [116].

The Likelihood Function and Distortion

Lemma 2.3.4. *Maximizing log likelihood also minimizes the weighted sum of squares*



(A) Fuzzy c-means, $m = 1.3$, $z\,axis = ln(d_{ij}^{-\frac{1}{m-1}})$

(B) Fuzzy c-means, $m = 1.3$, $z\,axis = \forall i, \max_{j=1}^{C}(U)$

(C) Soft k-means, $\beta = 0.4$, $z\,axis = \exp(-\beta d_{ij})$

(D) Soft k-means, $\beta = 0.4$, $z\,axis = \forall i, \max_{j=1}^{K}(R)$

FIGURE 2.11: The resemblance between fuzzy c-means and soft k-means.

Proof: For any random vector $\mathbf{x}$, we can write soft k-means as a superposition of K one-dimensional Gaussian distributions with equal variances and mixing proportions,

$$p(\mathbf{x}|\mathbf{Z}, \beta) = \frac{1}{K}\sum_{i=1}^{K} p(\mathbf{x}|\mathbf{z}_j, \beta), \text{ where} \tag{2.64}$$

$$p(\mathbf{x}|\mathbf{z}_j, \beta) = \frac{1}{\sqrt{\pi/\beta}} \exp\left(-\beta||\mathbf{x} - \mathbf{z}_j||^2\right), \tag{2.65}$$

With the above definition, we then observe the log likelihood for an individual Gaussian,

$$\log \mathcal{L}(\theta_j) = \log \prod_{i=1}^{N} p(\mathbf{x}_i|\mathbf{z}_j, \beta), \tag{2.66}$$

$$= \sum_{i=1}^{N} \log p(\mathbf{x}_i|\mathbf{z}_j, \beta), \tag{2.67}$$

$$= -\left[\overbrace{\sum_{i=1}^{N} \frac{1}{2}\Big(\log \pi - \log \beta \Big)}^{\text{constant}} + \beta \quad \overbrace{\sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{z}_j||^2}^{\text{weighted sum of squares}} \right]. \tag{2.68}$$

It is therefore obvious that maximizing $\log \mathcal{L}(\theta_j)$ is equivalent to minimizing the weighted sum of squares. ■

Expectation Step

Lemma 2.3.5 *(Optimality of the Expectation Step). Given $D(\mathbf{X}, \mathbf{Z})$, $f(U, D(\mathbf{X}, \mathbf{Z}), \beta)$ is optimized when*

$$u_{ij} = \frac{\exp\left(-\beta d_{ij}\right)}{\sum_{k=1}^{K} \exp\left(-\beta d_{ik}\right)} = \frac{\exp\left(-\beta ||\mathbf{x}_i - \mathbf{z}_j||^2\right)}{\sum_{k=1}^{K} \exp\left(-\beta ||\mathbf{x}_i - \mathbf{z}_k||^2\right)}. \tag{2.69}$$

Proof: We can compute the responsibility of the j^{th} Gaussian for describing an observation $\mathbf{x}_i$ using the conditional probability of $\mathbf{z}_j = 1$ given $\mathbf{x}_i$ and β. Using Bayes rule we obtain,

$$\begin{aligned} u_{ij} = p(\mathbf{z}_j = 1|\mathbf{x}_i, \beta) &= \frac{\frac{1}{K} p(\mathbf{x}_i|\mathbf{z}_j, \beta)}{\sum_{k=1}^{K} \frac{1}{K} p(\mathbf{x}_i|\mathbf{z}_k, \beta)} \\ &= \frac{\exp\left(-\beta ||\mathbf{x}_i - \mathbf{z}_j||^2\right)}{\sum_{k=1}^{K} \exp\left(-\beta ||\mathbf{x}_i - \mathbf{z}_k||^2\right)}. \end{aligned} \tag{2.70}$$

■

Maximization Step

Lemma 2.3.6 *(Optimality of the Maximization Step). Given U, $f(U, D(\mathbf{X}, \mathbf{Z}), \beta)$ is minimized when* $\mathbf{z}_j = \dfrac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}$

Proof: $f(U, D(\mathbf{X}, \mathbf{Z}), \beta)$ is minimized when the gradient w.r.t $\mathbf{z}_j : j \in \{1, \ldots, K\}$ vanish,

$$\begin{aligned}
\nabla_{\mathbf{z}_j} f(U, D(\mathbf{X}, \mathbf{Z}), \beta) &= \nabla_{\mathbf{z}_j} \sum_{j=1}^{K} \sum_{i=1}^{N} u_{ij} d_{ij}, \\
&= \nabla_{\mathbf{z}_j} \sum_{j=1}^{K} \sum_{i=1}^{N} u_{ij} (\mathbf{x}_i - \mathbf{z}_j)^T (\mathbf{x}_i - \mathbf{z}_j), \\
&= -2 \sum_{j=1}^{K} \sum_{i=1}^{N} u_{ij} (\mathbf{x}_i - \mathbf{z}_j), \\
&= -2 \sum_{j=1}^{K} \left(\sum_{i=1}^{N} u_{ij} \mathbf{x}_i - \sum_{i=1}^{N} u_{ij} \mathbf{z}_j \right) \qquad \doteq \quad 0.
\end{aligned} \tag{2.71}$$

It then follows that solving $\mathbf{z}_j : j \in \{1, \ldots, K\}$ proves Lemma 2.3.6,

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}. \quad \blacksquare \tag{2.72}$$

2.3.5 Gaussian Mixture Models

Gaussian Mixture Models (GMM) [116] can be seen as a probabilistically *complete* formulation for the soft k-means. The K fuzzy partitions in GMM can be probabilistically described as a mixture of K multivariate Gaussians such that for a random vector $\mathbf{x} \in \mathbb{R}^{dim}$ we have,

$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{K} \alpha_j p(\mathbf{x}|\mathbf{z}_j, \theta_j), \quad \Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}, \tag{2.73}$$

$$p(\mathbf{x}|\theta_j) = p(\mathbf{x}|\mathbf{z}_j, \Sigma_j) \tag{2.74}$$

$$= \frac{1}{\sqrt{(2\pi)^{dim}|\Sigma_j|}} \exp\left(-\frac{(\mathbf{x} - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{z}_j)}{2} \right). \tag{2.75}$$

where $|\Sigma_j|$ denotes the determinant of Σ_j, for each $j : j \in \{1, \ldots, K\}, p(\mathbf{x}_i|z_j, \theta_j)$ are mixture components, each are multivariate Gaussian probability density function defined by a $dim \times 1$ centroid vector and a $dim \times dim$ covariance matrix, such that $\theta_j = \{\mathbf{z}_j, \Sigma_j\}$. α_j denotes the mixing proportion where $\alpha_j = p(\mathbf{z}_j)$, $\sum_{j=1}^{K} \alpha_j = 1$. α_j represents the probability that $\mathbf{x}$ was generated by the j^{th} mixture. All these parameters are stored in the set of parameters Θ.

The parameters Θ are optimized using the EM method, as follows:

Expectation: The responsibility that a j^{th} Gaussian density takes for describing an observation vector $\mathbf{x}_i$ can be derived simply as the conditional probability of the cluster $\mathbf{z}_{ij}$ given an observation $\mathbf{x}_i$ and a set of parameters Θ. Using Bayes rule, we can derive u_{ij} as follows,

$$\begin{aligned} u_{ij} &= p(\mathbf{z}_j|\mathbf{x}_i, \Theta), \\ &= \frac{\alpha_j p(\mathbf{x}_i|\theta_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\theta_k)}. \end{aligned} \tag{2.76}$$

It must then follow that $\forall i, \sum_{j=1}^{K} u_{ij} = 1$.

Maximization: Given the membership matrix U, for each $j : j \in \{1, \ldots, K\}$, α_j, $\mathbf{z}_j$, and Σ_j, can be updated as follows,

1. Calculate the updated mixing proportions $\hat{\alpha}_{\{j,\ldots,K\}}$,

$$\hat{\alpha}_j = \frac{1}{N} \sum_{i=1}^{N} u_{ij}, \tag{2.77}$$

2. Given $\hat{\alpha}_{\{j,\ldots,K\}}$, update the centroid vectors $\mathbf{z}_{\{j,\ldots,K\}}$ using the weighted average $\forall j, \mathbf{x} \in \mathbb{C}_{z_j}$,

$$\hat{\mathbf{z}}_j = \frac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}, \tag{2.78}$$

3. Given $\hat{\mathbf{z}}_{\{j,\ldots,K\}}$ and $\hat{\alpha}_{\{j,\ldots,K\}}$, update the covariance matrices using the weighted average $\forall j, \Sigma_j \in \mathbb{C}_{z_j}$,

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^{N} u_{ij} (\mathbf{x}_i - \hat{\mathbf{z}}_j)(\mathbf{x}_i - \hat{\mathbf{z}}_j)^T}{\sum_{i=1}^{N} u_{ij}}. \tag{2.79}$$

EM algorithm is a Maximum Likelihood Estimator (MLE). It maximizes the log likelihood of the model parameters given the data, $\mathcal{L}(\Theta|\mathbf{X})$, as follows,

$$\hat{\Theta}_{MLE} = \arg\max_{\Theta} \Big\{ \log \mathcal{L}(\Theta|\mathbf{X}) \Big\}, \tag{2.80}$$

where in the context of GMM, the likelihood function is as follows,

$$\begin{aligned}\log \mathcal{L}(\Theta|\mathbf{X}) &= \log \prod_{i=1}^{N} p(\mathbf{x}_i|\Theta) = \sum_{i=1}^{N} \log p(\mathbf{x}_i|\Theta), \\ &= \sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j).\end{aligned} \tag{2.81}$$

Convergence of the EM algorithm can be detected when the improvement of $\log \mathcal{L}(\Theta|\mathbf{X})$ is minimal.

The set of initial parameters for the EM algorithm can be chosen at random or via heuristic method (e.g. k-means/fuzzy c-means/soft k-means).

The result on five Gaussians dataset can be seen in Figure 2.12. The pseudocode for the GMM-EM algorithm can be seen in Algorithm 2.5

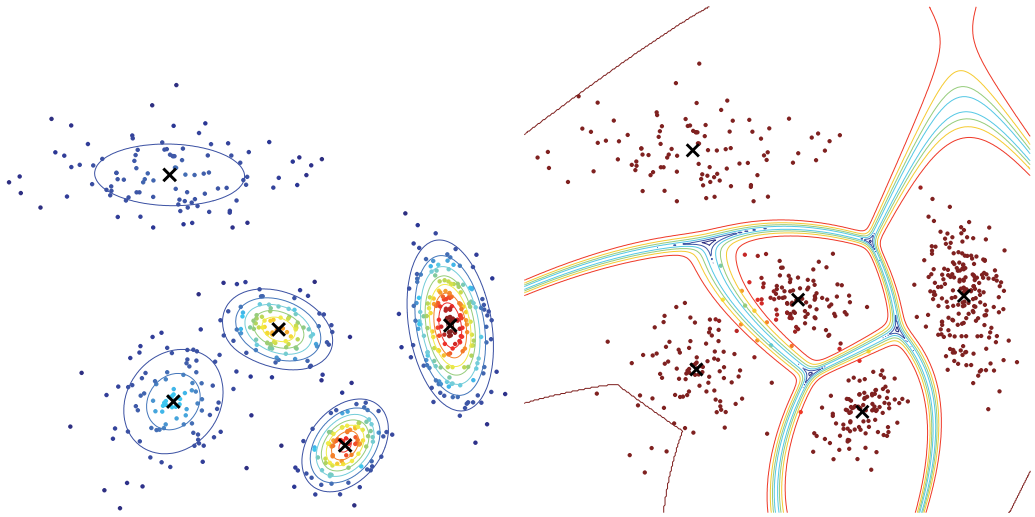
Algorithm 2.5 GMM pseudocode

Input: $\mathbf{X}$, # mixtures K, max iteration t_{max}, and gradient threshold ε.
Output: Gaussian mixtures $p(\mathbf{x}|\Theta)$ and fuzzy membership matrix U.

1: Initialize Θ using { random/k-means/fuzzy c-means/soft k-means },
2: **repeat**
3: **Expectation:**
4: $\forall i, j : u_{ij} \leftarrow \dfrac{\alpha_j p(\mathbf{x}_i|\theta_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\theta_k)}$,
5: **Maximization:**
6: $\forall j, \alpha_j \leftarrow \dfrac{\sum_{i=1}^{N} u_{ij}}{N}$,
7: $\forall j, \mathbf{z}_j \leftarrow \dfrac{\sum_{i=1}^{N} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}$,
8: $\forall j, \Sigma_k \leftarrow \dfrac{\sum_{i=1}^{N} u_{ij}(\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^T}{\sum_{i=1}^{N} u_{ij}}$,
9: **until** $-\partial \log \mathcal{L}(\Theta|\mathbf{X})/\partial t < \varepsilon \ \vee \ (t > t_{max})$

2.3.5.1 Proof of Convergence

The EM algorithm generally aims to maximize the likelihood function of the data given the model [118]. As a general definition, GMM sought to model the overall distribution of the data as a sum



(A) $p(\mathbf{x}|\Theta)$ (B) $\forall i, \max_{j=1}^{C}(U)$

FIGURE 2.12: GMM results on the Five Gaussians dataset.

of Gaussians (refer to Equation 2.73). The likelihood function is described as follows,

$$\begin{aligned} l(\Theta) = \log \mathcal{L}(\Theta|\mathbf{X}) &= \log \prod_{i=1}^{N} p(\mathbf{x}_i|\Theta) = \sum_{i=1}^{N} \log p(\mathbf{x}_i|\Theta), \\ &= \sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j), \\ &= l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}|\mathbf{X}), \end{aligned} \tag{2.82}$$

which can be maximized by solving Θ given $\mathbf{X}$ and K are known. However, the optimal value for Θ cannot be found analytically and must be estimated using the EM algorithm [118–120]. Note that the parameters to be optimized Θ consists of:

1. mixing proportions $\mathcal{A}$,
2. centroid vectors $\mathbf{Z}$, and
3. covariance matrices $\mathbf{\Sigma}$.

Hence in order to prove the convergence of the EM algorithm for GMM, we need to show that optimizing each component guarantees the improvement of $\mathcal{L}(\Theta)$.

Derivations in this subsection are personally done by the author based on the work of Xu and Jordan [120]. Bishop's Pattern Recognition and Machine Learning is also used as an important reference pertaining the concepts and methods [116].

Updating the Mixing Coefficients

Lemma 2.3.7 *(Convergence of EM algorithm for GMM (1)). For each iteration of the EM algorithm,* $l(\hat{\Theta}) > l(\Theta)$ *when* $\hat{\alpha}_j = \frac{1}{N}\sum_{i=1}^{N} u_{ij}$

Proof: The updated mixing coefficients $\hat{\mathcal{A}}$ maximizes $l(\Theta)$ when the partial derivative of $\partial l(\Theta)/\partial \mathcal{A} = 0$ subject to the constraint $\sum_{j=1}^{K} \alpha_j$. We can then express this lemma as a constrained optimization problem,

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}|\mathbf{X}) = \sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j), \\ \text{s.t.} \quad & \alpha_j \in \{0, 1\}, \\ \text{s.t.} \quad & \sum_{j=1}^{K} \alpha_j \ = \ 1, \end{aligned} \tag{2.83}$$

which can be simplified by introducing the Lagrange multiplier λ as follows,

$$l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}, \lambda|\mathbf{X}) = \left(\sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) \right) + \lambda \left(1 - \sum_{j=1}^{K} \alpha_j \right). \tag{2.84}$$

Setting the partial derivative with respect to $\mathcal{A}$ to 0 we have

$$\begin{aligned} \frac{\partial l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}, \lambda|\mathbf{X})}{\partial \mathcal{A}} = \sum_{i=1}^{N} \frac{p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} - \lambda = 0, \\ \lambda = \sum_{i=1}^{N} \frac{p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)}. \end{aligned} \tag{2.85}$$

Multiplying both sides of Equation 2.85 with α_j gives

$$\alpha_j = \frac{1}{\lambda} \sum_{i=1}^{N} \frac{\alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} = \frac{1}{\lambda} \sum_{i=1}^{N} u_{ij}, \tag{2.86}$$

Imposing the constraint $\sum_{j=1}^{K} \alpha_j \ = \ 1$ to Equation 2.86 we get,

$$\begin{aligned} \frac{1}{\lambda} \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij} = 1 \\ \lambda = N. \end{aligned} \tag{2.87}$$

It then follows that substituting Equation 2.87 to Equation 2.86 gives the first proof,

$$\hat{\alpha}_j = \frac{1}{N}\sum_{i=1}^{N} u_{ij}. \quad \blacksquare \tag{2.88}$$

Updating the Cluster Centres

Lemma 2.3.8 *(Convergence of EM algorithm for GMM (2)).* ***For each iteration of the EM algorithm,*** $l(\hat{\Theta}) > l(\Theta)$ ***when*** $\hat{\mathbf{z}}_j = \dfrac{\sum_{i=1}^{N} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}$

Proof: The updated centroid vectors $\hat{\mathbf{Z}}$ maximizes $l(\Theta)$ when the partial derivative of $\partial l(\Theta)/\partial \mathbf{Z} = 0$,

$$\begin{aligned}
l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}|\mathbf{X}) &= \sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j), \\
\frac{\partial l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}|\mathbf{X})}{\partial \mathbf{Z}} &= \sum_{i=1}^{N} \frac{1}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} \frac{\partial \left(\sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j)\right)}{\partial \mathbf{Z}}, \\
&= -\sum_{i=1}^{N}\sum_{j=1}^{K} \frac{\alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j)}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} \Sigma_j^{-1}(\mathbf{x}_i - \hat{\mathbf{z}}_j) = 0, \\
&= -\sum_{i=1}^{N}\sum_{j=1}^{K} u_{ij}\Sigma_j^{-1}(\mathbf{x}_i - \hat{\mathbf{z}}_j) = 0,
\end{aligned} \tag{2.89}$$

which consequently implies that $\dfrac{\partial l(\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}|\mathbf{X})}{\partial \mathbf{Z}} = 0$ when each Gaussian mixture is optimized. As covariance matrices are positive definite, it then follows that multiplying both sides with Σ_j and rearranging gives the second proof,

$$\begin{aligned}
-\sum_{i=1}^{N} u_{ij}\Sigma_j^{-1}(\mathbf{x}_i - \hat{\mathbf{z}}_j) &= 0, \\
\sum_{i=1}^{N} u_{ij}\hat{\mathbf{z}}_j &= \sum_{i=1}^{N} u_{ij}\mathbf{x}_i, \\
\hat{\mathbf{z}}_j &= \frac{\sum_{i=1}^{N} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}. \quad \blacksquare
\end{aligned} \tag{2.90}$$

Updating the Covariance Matrices

Lemma 2.3.9 *(Convergence of EM algorithm for GMM (3)).* ***For each iteration of the EM algorithm,*** $l(\hat{\Theta}) > l(\Theta)$ ***when*** $\hat{\Sigma}_k = \dfrac{\sum_{i=1}^{N} u_{ij}(\mathbf{x}_i - \hat{\mathbf{z}}_j)(\mathbf{x}_i - \hat{\mathbf{z}}_j)^T}{\sum_{i=1}^{N} u_{ij}}$

Proof: For each multivariate Gaussian mixture, the updated covariance matrix $\forall j, \hat{\Sigma}_j$ maximizes $l(\Theta)$ when the partial derivative with respect to $\forall j, \Sigma_j$ vanishes,

$$\begin{aligned}
l(\mathbf{X}|\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma}) &= \sum_{i=1}^{N} \log \sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j), \\
\frac{\partial l(\mathbf{X}|\mathcal{A}, \mathbf{Z}, \mathbf{\Sigma})}{\partial \mathbf{\Sigma}} &= \sum_{i=1}^{N} \frac{1}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} \frac{\partial \left(\sum_{j=1}^{K} \alpha_j p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) \right)}{\partial \mathbf{\Sigma}}, \\
&= \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{\alpha_j}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} \frac{\partial \left(p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) \right)}{\partial \Sigma_j} = 0, \\
&= \sum_{j=1}^{K} \frac{\partial l(\mathbf{X}|\alpha_j, \mathbf{z}_j, \Sigma_j)}{\partial \Sigma_j} = 0, \text{where} \\
\frac{\partial l(\mathbf{X}|\alpha_j, \mathbf{z}_j, \Sigma_j)}{\partial \Sigma_j} &= \sum_{i=1}^{N} \frac{\alpha_j}{\sum_{k=1}^{K} \alpha_k p(\mathbf{x}_i|\mathbf{z}_k, \Sigma_k)} \frac{\partial \left(p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) \right)}{\partial \Sigma_j} = 0.
\end{aligned} \tag{2.91}$$

As $\forall j, \Sigma_j$ is a positive definite matrix, as to matrix algebra [121] we have by definition,

$$\frac{\partial |\Sigma_j|}{\partial \Sigma_j} = |\Sigma_j|(\Sigma_j^{-1})^T = |\Sigma_j|(\Sigma_j^{-1}), \quad \text{and} \tag{2.92}$$

$$\frac{\partial (\mathbf{x}_i - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mathbf{z}_j)}{\partial \Sigma_j} = -\left(\Sigma_j^{-1} (\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^T \Sigma_j^{-1} \right)^T = -\Sigma_j^{-1} \mathfrak{N}_{ij} \Sigma_j^{-1}, \tag{2.93}$$

where $\mathfrak{N}_{ij} = (\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^T$ as noted in Equation 2.93.

Based on Equation 2.92 and Equation 2.93 We can then rewrite

$$\begin{aligned}
\frac{\partial \left(p(\mathbf{x}_i|\mathbf{z}_j, \Sigma_j) \right)}{\partial \Sigma_j} &= (2\pi)^{-dim/2} \Bigg[-\frac{1}{2} |\Sigma_j|^{-3/2} |\Sigma_j| \Sigma_j^{-1} \exp\left(-\frac{(\mathbf{x} - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{z}_j)}{2} \right) \\
&\qquad + \frac{1}{2} |\Sigma_j|^{-1/2} \exp\left(-\frac{(\mathbf{x} - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{z}_j)}{2} \right) \Sigma_j^{-1} \mathfrak{N}_{ij} \Sigma_j^{-1} \Bigg], \\
&= -\frac{1}{2} (2\pi)^{-dim/2} |\Sigma_j|^{-1/2} \exp\left(-\frac{(\mathbf{x} - \mathbf{z}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{z}_j)}{2} \right) \left[\Sigma_j^{-1} - \Sigma_j^{-1} \mathfrak{N}_{ij} \Sigma_j^{-1} \right], \\
&= -\frac{1}{2} p(\mathbf{x}_i|\mathbf{z}_j \Sigma_j) \left(\Sigma_j^{-1} \left[1 - \mathfrak{N}_{ij} \Sigma_j^{-1} \right] \right).
\end{aligned} \tag{2.94}$$

Substituting Equation 2.94 back to Equation 2.91 we obtain for all j,

$$\begin{aligned}
-\frac{1}{2}\sum_{i=1}^{N}\frac{\alpha_j p(\mathbf{x}_i|\mathbf{z}_j\Sigma_j)}{\sum_{k=1}^{K}\alpha_k p(\mathbf{x}_i|\mathbf{z}_k,\Sigma_k)}\left(\hat{\Sigma}_j^{-1}\left[1-\mathfrak{N}_{ij}\hat{\Sigma}_j^{-1}\right]\right) &= 0,\\
\sum_{i=1}^{N}u_{ij}\left(\hat{\Sigma}_j^{-1}\left[1-\mathfrak{N}_{ij}\hat{\Sigma}_j^{-1}\right]\right) &= 0,\\
\sum_{i=1}^{N}u_{ij}-\sum_{i=1}^{N}u_{ij}\mathfrak{N}_{ij}\hat{\Sigma}_j^{-1} &= 0,\\
\sum_{i=1}^{N}u_{ij} &= \sum_{i=1}^{N}u_{ij}\mathfrak{N}_{ij}\hat{\Sigma}_j^{-1},\\
\hat{\Sigma}_j &= \frac{\sum_{i=1}^{N}u_{ij}\mathfrak{N}_{ij}}{\sum_{i=1}^{N}u_{ij}},\\
\hat{\Sigma}_j &= \frac{\sum_{i=1}^{N}u_{ij}(\mathbf{x}_i-\mathbf{z}_j)(\mathbf{x}_i-\mathbf{z}_j)^T}{\sum_{i=1}^{N}u_{ij}}. \quad \blacksquare
\end{aligned} \tag{2.95}$$

EM Algorithm and Gaussian Mixtures: The Corollary

Corollary 2.3.1 *(Convergence of EM algorithm for GMM).* *We have proven for Lemma 2.3.7, Lemma 2.3.8, and Lemma 2.3.9 that* $\mathcal{L}(\hat{\Theta}) > \mathcal{L}(\Theta)$ *when*

$$\hat{\alpha}_j = \frac{1}{N}\sum_{i=1}^{N}u_{ij},$$

$$\hat{\mathbf{z}}_j = \frac{\sum_{i=1}^{N}u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N}u_{ij}}, \quad \textit{and}$$

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^{N}u_{ij}(\mathbf{x}_i-\hat{\mathbf{z}}_j)(\mathbf{x}_i-\hat{\mathbf{z}}_j)^T}{\sum_{i=1}^{N}u_{ij}}.$$

By alternating between the Expectation and Maximization steps, we can then guarantee that EM will converge to the local optimum given any initial set of parameters.

2.4 Cluster Validity Indices

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Jain and Dubes [122].

Validation of a clustering result refers to the problem on how to validate the goodness of the resulting partition given the data [123]. As clustering is inherently a non-deterministic process, there are chances where the results can be absolutely counter-intuitive when assumptions made prior to clustering are incorrect. At times incomplete deduction can be made simply because of suboptimal results. Mistakes may arise from various reasons including:

- incorrect specifications e.g. number of clusters/β/m, inappropriate distance measure, or simply bad initializations;
- incorrect features/feature representations which may result in counter-intuitive partitions, even after numerous trials;
- the shapes of clusters in the feature space may be non-convex/non-gaussian by nature.

The limitations of divisive clustering have been made clear in the previous section. Acknowledging this, cluster validity indices are designed under an assumption that *if a population can be represented by Gaussian mixtures, an optimum compromise solution should exist given the correct number of clusters (K or C) is specified.* Various validity measures have been proposed in the literature, each based on differing assumptions and definitions on how a sufficiently *good* clustering should look like. However, as to clustering algorithms, validity index is not a "panacea" to clustering problems. For example, there is no way to objectively validate non-convex clusters with prototype-based measures that assumes convexity. Validating the goodness of a cluster is, and always will be, an illusive process [122]. This chapter will be devoted to briefly review a number of widely used internal and external cluster validity indices available in the literature.

2.4.1 Internal Validity Indices

Internal validity indices measure the quality of a clustering result based on the intrinsic information available in the data itself. The methods described in this section covers only a small portion of the available methods in the literature.

Definition 2.4.1 *(Partition Coefficient).* Partition coefficient measures the amount of overlapping between clusters [124],

$$\mathfrak{V}_{PC} = \frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{N} u_{ij}^2, \tag{2.96}$$

where u_{ij} is the membership of data point i in cluster j. A major criticism on $\mathfrak{V}_{PC}$ is that it lacks direct connection to the geometrical property of the data [125, 126]. The optimal number of clusters is indicated when $\mathfrak{V}_{PC}$ is maximized.

Definition 2.4.2 *(Cluster Entropy).* The cluster entropy was introduced by Bezdek in 1975 [87]. Silva et al. have shown the direct correlation between the decrease in Shannon Entropy and the improvement of a fuzzy model [127]. Shannon entropy is maximized at $p(x) = 0.5$; and minimized at $p(x) = 1.0$ and $p(x) \to 0^+$ [89]. The formulation is given as follows,

$$H(p(x)) = -p(x)\log p(x), \tag{2.97}$$

which, in the context of cluster validity [87], can be expressed as the conditional probability of the cluster given the data, normalized to the number of data such that

$$\mathfrak{V}_H = \frac{1}{N}H(p(\Theta|\mathbf{X})), \tag{2.98}$$

$$= -\frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{N} p(\mathbf{x}_i|\theta_j)\log p(\mathbf{x}_i|\theta_j), \tag{2.99}$$

$$= -\frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{N} u_{ij}\log u_{ij}. \tag{2.100}$$

Similarly to $\mathfrak{V}_{PC}$, $\mathfrak{V}_{CE}$ receives similar criticism due to its solely probabilistic nature.

Definition 2.4.3 *(Partition Index).* Bensaid et al. proposes the Partition Index $\mathfrak{V}_{SC}$ which takes into account both the fuzzy memberships and the structure of the data [128]. $\mathfrak{V}_{SC}$ defines a good clustering as compact and well separated [128]. Compactness π_j and separation s_j of a cluster $\mathbb{C}_j$ can be computed as follows,

$$\pi_j = \frac{\sum_{i=1}^{N} u_{ij}^m (\mathbf{x}_i - \mathbf{z}_j)^T \Sigma^{-1} (\mathbf{x}_i - \mathbf{z}_j)}{\sum_{i=1}^{N} u_{ij}}, \tag{2.101}$$

$$s_j = \sum_{k=1}^{K} (\mathbf{z}_j - \mathbf{z}_k)\Sigma^{-1}(\mathbf{z}_j - \mathbf{z}_k), \tag{2.102}$$

where Σ is a positive definite matrix which denotes the covariance of the distance measure and $m \in [1, \infty)$ is a constant which determines the fuzziness of the partitions. The validity index is simply the ratio over all clusters,

$$\mathfrak{V}_{SC} = \sum_{j=1}^{K} \frac{\pi_j}{s_j} = \sum_{j=1}^{K} \left[\frac{\sum_{i=1}^{N} u_{ij}^m (\mathbf{x}_i - \mathbf{z}_j)^T \Sigma^{-1} (\mathbf{x}_i - \mathbf{z}_j)}{\left(\sum_{i=1}^{N} u_{ij} \right) \left(\sum_{k=1}^{K} (\mathbf{z}_j - \mathbf{z}_k) \Sigma^{-1} (\mathbf{z}_j - \mathbf{z}_k) \right)} \right], \tag{2.103}$$

where lower $\mathfrak{V}_{SC}$ indicates a better partition.

Definition 2.4.4 *(Silhouette Width Criterion).* The Silhouette takes into account geometrical considerations about compactness and separation of clusters [129]. The silhouette of a point can be described as the compromise between the average dissimilarity between an object $\mathbf{x}_i$ with all other objects its own cluster ($a(i)$) and its closest neighboring cluster ($b(i)$). The Silhouette width criterion $\mathfrak{V}_{SWC}$ can be calculated simply by taking the average silhouette value over all points. The formulation is as follows,

$$a(i) = \frac{1}{|\mathbb{C}_z|} \sum_{\mathbf{x}_i, \mathbf{y} \in \mathbb{C}_z} d(\mathbf{x}_i, \mathbf{y}) \tag{2.104}$$

$$b(i) = \min_{\substack{t \in \{1,\ldots,K\} \\ s \neq t}} \left\{ \frac{1}{|\mathbb{C}_t|} \sum_{\mathbf{x}_i \in \mathbb{C}_s, \mathbf{y} \in \mathbb{C}_t} d(\mathbf{x}_i, \mathbf{y}) \right\} \tag{2.105}$$

$$s_i = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{2.106}$$

$$\mathfrak{V}_{SWC} = \frac{1}{N} \sum_{i=1}^{N} s_i. \tag{2.107}$$

where the denominator is simply a normalizer such that $s_j, \mathfrak{V}_{SWC} \in \{-1, 1\}$.

Definition 2.4.5 *(Simplified Silhouette Width Criterion).* The scalability issue in the Silhouette width criterion is apparent due to the requirement of the pairwise distance computation in both $a(i)$ and $b(i)$. To simplify the computation of s_i to linear complexity, Vendramin et al. proposes

the following modification,

$$p(i) = d(\mathbf{x}_i, \mathbf{z}_j)_{\mathbf{x}_i \in \mathbb{C}_j}, \tag{2.108}$$

$$q(i) = \min_{\substack{t \in \{1,\dots,K\} \\ t \neq j}} \left\{ d(\mathbf{x}_i, \mathbf{z}_t) \right\}_{\mathbf{x}_i \notin \mathbb{C}_t}, \tag{2.109}$$

$$ss_i = \frac{p(i) - q(i)}{\max\{p(i), q(i)\}}, \tag{2.110}$$

$$\mathfrak{V}_{SSWC} = \frac{1}{N} \sum_{i=1}^{N} ss_i. \tag{2.111}$$

hence reducing the Silhouette complexity to linear time/space. $\mathfrak{V}_{SSWC}$ close to 1 indicates better clustering.

Definition 2.4.6 *(Dunn Index).* The Dunn Index [115] is proposed as follows,

$$\mathfrak{V}_{DN} = \min_{\substack{t \in \{1,\dots,K\} \\ s \in \{1,\dots,K\} \\ s \neq t}} \left\{ \frac{\delta\left(\mathbb{C}_s, \mathbb{C}_t\right)}{\max\limits_{r \in \{1,\dots,K\}} \{\Delta(\mathbb{C}_r)\}} \right\}, \tag{2.112}$$

where $\delta(\mathbb{C}_s, \mathbb{C}_t)$ denotes the (minimum) *set distance* between $\mathbb{C}_s$ and $\mathbb{C}_t$, while $\Delta(\mathbb{C}_r)$ denotes the (maximum) diameter of $\mathbb{C}_r$, as follows,

$$\delta(\mathbb{C}_s, \mathbb{C}_t) = \min_{\substack{\mathbf{x} \in \mathbb{C}_s \\ \mathbf{y} \in \mathbb{C}_t}} d(\mathbf{x}, \mathbf{y}) = \delta_{SL}(\mathbb{C}_s, \mathbb{C}_t), \tag{2.113}$$

$$\Delta(\mathbb{C}_r) = \max_{\mathbf{x}, \mathbf{y} \in \mathbb{C}_r} d(\mathbf{x}, \mathbf{y}). \tag{2.114}$$

Note the way δ is formulated resembles the single linkage distance function, while Δ resembles the maximum intracluster distance.

Definition 2.4.7 *(Generalized Dunn Index).* Bezdek and Pal proposed that both δ and Δ makes Dunn index overly sensitive to noises and subtle changes in the data geometry [130]. Bezdek and Pal proposed to use other linkage functions, one of them resembles the centroid linkage metric (ZL) as follows,

$$\delta_{ZL}\left(\mathbb{C}_s, \mathbb{C}_t\right) = \frac{1}{|\mathbb{C}_s| + |\mathbb{C}_t|} \left[\sum_{\mathbf{y} \in \mathbb{C}_s} d(\mathbf{y}, \mathbf{z}_t) + \sum_{\mathbf{y} \in \mathbb{C}_t} d(\mathbf{y}, \mathbf{z}_s) \right], \tag{2.115}$$

$$\Delta_{ZL}(\mathbb{C}_r) = \frac{2}{|\mathbb{C}_r|} \sum_{\mathbf{y} \in \mathbb{C}_r} d(\mathbf{y}, \mathbf{z}_r), \tag{2.116}$$

where $\delta_{ZL}(\mathbb{C}_s, \mathbb{C}_t)$ denotes the average pairwise distances between elements and centroids in clustered sets $\mathbb{C}_s$ and $\mathbb{C}_t$. $\Delta_{ZL}(\mathbb{C})_r$ denotes the average diameter of the clustered set $\mathbb{C}_r$. From this

point on, we will refer the generalized Dunn Index as $\mathfrak{V}_{DN(ZL)}$ as follows,

$$\mathfrak{V}_{DN(ZL)} = \min_{\substack{t\in\{1,\ldots,K\}\\ s\in\{1,\ldots,K\}\\ s\neq t}} \left\{ \frac{\delta_{ZL}\left(\mathbb{C}_s, \mathbb{C}_t\right)}{\max\limits_{r\in\{1,\ldots,K\}} \{\Delta_{ZL}(\mathbb{C}_r)\}} \right\}. \tag{2.117}$$

Definition 2.4.8 *(Adjusted Rand Index).* The adjusted rand index [131] measures the agreement between elements in clustered set $\mathbb{C}$ and elements in target set $\mathbb{T}$ [132]. The Adjusted Rand Index is calculated as follows:

$$\mathfrak{V}_{Adj.Rand} = \frac{\binom{n_j}{2}(a+d) - ((a+b)(a+c)+(c+d)(b+d))}{\binom{n_j}{2}^2 - ((a+b)(a+c)+(c+d)(b+d))}, \tag{2.118}$$

where:

- a: number of element pairs belong to the same set in both $\mathbb{C}$ and $\mathbb{T}$,
- b: number of element pairs that belong to the same set in $\mathbb{C}$ but different sets in $\mathbb{T}$,
- c: number of element pairs that belong to different sets in $\mathbb{C}$ but the same set in $\mathbb{T}$, and
- d: number of element pairs that belong to different sets in both $\mathbb{C}$ and $\mathbb{T}$.

An adjusted rand index of one indicates a perfect agreement, zero suggests agreement due to chance, while negative indicates agreement less than chance [132].

Definition 2.4.9 *(Calinski-Harabasz Index).* The Calinski-Harabasz (CH) index [133] measures clustering quality based on the traces of between-cluster and within-cluster distance scatter matrices. CH index is formulated as follows:

$$\begin{aligned}
\mathbf{m} &= \frac{\sum_{j=1}^{n_j} \mathbf{y}_j}{n_j}, \\
Tr(\mathrm{S}_B) &= \sum_{i=1}^{n_c} n_i d(\mathbf{x}_i, \mathbf{m}), \\
Tr(\mathrm{S}_W) &= \sum_{i=1}^{n_c}\sum_{j=1}^{n_i} d(\mathbf{y}_j, \mathbf{x}_i), \\
\mathfrak{V}_{CH} &= \frac{Tr(\mathrm{S}_B)/(n_c-1)}{Tr(\mathrm{S}_W)/(n_j-n_c)},
\end{aligned} \tag{2.119}$$

A large value of CH suggests a clustering result with good quality [132].

2.4.2 External Validity Indices

External validity indices measure the quality of a clustering result based on an available external label information based on the domain/previous knowledge. They are powerful tools to investigate the coherence of a clustering when a valid reference is readily available. However, the practicality of external validation indices are limited, due to the fact that prior information of the dataset in question is either not available or under continuous investigation in many practical cases.

Definition 2.4.10 *(Classification Entropy)***.** Classification Entropy tests the cluster homogeneity against a reference class label vector. Lower classification entropy indicates that objects in the database are homogeneous. Homogeneous clusters are less likely to be misclassified. Smaller classification entropy indicates homogeneity, and therefore better clustering.

For each cluster j we compute p_{rj}, the probability of the member of the cluster j belongs to the class r as follows,

$$p_{rj} = \frac{n_{rj}}{n_j}, \tag{2.120}$$

where n_j is the number of elements in the cluster j, and n_{rj} is the number of class r in the cluster j. The entropy of the j^{th} cluster can be calculated,

$$H_j = -\sum_{r=1}^{C} p_{rj} \log p_{rj}, \tag{2.121}$$

where C denotes the number of classes provided in the reference label vector. The total entropy can be calculated simply as the weighted sum of the entropy of each cluster H_j normalized by the number of data vectors in the problem set,

$$\mathfrak{V}_E = \frac{1}{N} \sum_{j=1}^{K} n_j H_j, \tag{2.122}$$

where n_j denotes the number of elements in the cluster j, N denotes the total number of elements in the problem set, K denotes the number of clusters.

Definition 2.4.11 *(Purity)***.** Purity tests the quality of the clustering result against the reference class label vector. It is used to give an indication of the purity of the cluster by taking the ratio of the dominant class of the group in relation to the total number of objects inside the group. High purity is desirable for a good cluster.

For each cluster j we compute p_{rj}, the probability of the member of the cluster j belongs to the class r as follows,

$$p_{rj} = \frac{n_{rj}}{n_j}, \tag{2.123}$$

where n_j is the number of elements in the cluster j, and n_{rj} is the number of class r in the cluster j. The purity of the j^{th} cluster is calculated simply as,

$$purity_j = \max p_{rj}. \tag{2.124}$$

The total purity can be calculated simply as the weighted sum of the purity of each cluster $purity_j$ normalized by the number of data vectors in the problem set,

$$\mathfrak{V}_P = \frac{1}{N}\sum_{j=1}^{K} n_j purity_j, \tag{2.125}$$

where n_j denotes the number of elements in the cluster j, N denotes the total number of elements in the problem set, K denotes the number of clusters.

Definition 2.4.12 *(Percent Misclassified).* The percent misclassified is the ratio of false positive classifications to the number of objects,

$$\mathfrak{V}_{Pm} = \frac{1}{N}\sum_{j=1}^{K} n_j(1 - purity_j), \tag{2.126}$$

$$= 1 - \mathfrak{V}_P. \tag{2.127}$$

Definition 2.4.13 *(Normalized Mutual Information).* Investigating the Normalized Mutual information between the cluster labels U and reference class labels R can be a useful indication of the coherence between the clustering result against a provided reference. The formulation is as follows,

$$\begin{aligned} \mathfrak{V}_{NMI} &= NMI(\mathsf{U},\mathsf{R}) \\ &= \frac{I(\mathsf{U};\mathsf{R})}{\sqrt{H(\mathsf{U},\mathsf{R})}} \end{aligned} \tag{2.128}$$

2.5 Graph Theoretic Clustering

Graph theoretic clustering has been one of the major prerequisites in modern data analysis. Extensive practical usage of graph clustering is apparent in the analysis online socil networks [134–136] including Facebook, MySpace, LiveJournal, Youtube; Search engines [137] including Google PageRank [138], Bing and Yahoo [139]; Natural language processing; Financial networks and market analysis [140]; Complex biological networks [141]; and Image segmentation [90]. Graph clustering is an appropriate tool when the relationship between entities are either directly available or can be obtained without significant computation issue.

2.5.1 Graph Preliminaries

2.5.1.1 Graph Theoretic Definitions

Definition 2.5.1 *(Graph).* A graph is an entity which consists of interconnected vertices V and edges E, where — in the graph theoretic notation — can be written as $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$. $\mathbf{V}_{[N\times N]}$ is the vertex matrix which stores each vertex coordinate vector $\mathbf{v} \in \mathbb{V}$ in $\mathbb{R}^{dim}$; $\mathbf{E}_{[N\times N]}$ is the edge matrix which stores the edge affinity information between each pair of vertices, $E \in \mathbb{E}$ in $\mathbb{R}_{\geq 0}$.

Edge affinity matrix is inversely related to the pairwise proximity matrix (e.g. high affinity between patterns denotes higher similarity and vice versa). Many literatures refers edge affinity as the weight such that $w_{vw} = E_{vw}$, where 0 indicates no connection.

Another type of graph is the *unweighted graph* where edge connectivity is assigned as a boolean value $w_{vw} = [0, 1]$ where 1 indicates connection while 0 indicates the otherwise.

Graphs can be either directed or undirected. In directed graphs, for $(v, w) \in \mathbb{V}$, the edge $v \to w$ is the reversal of $w \to v$ and vice versa. In undirected graph direction does not matter, hence edge matrix is always symmetrical positive definite.

Definition 2.5.2 *(Subgraph).* $\mathbb{G}$ can be partitioned into disjoint subgraphs $\mathbb{C}_{\{1,\dots,K\}}$ where $\bigcup_{k=1}^{K} \mathbb{C}_k = \mathbb{G}$ and $\bigcap_{k=1}^{K} \mathbb{C}_k = \emptyset$.

Definition 2.5.3 *(Degree of a vertex).* For each vertex $V \in \mathbb{V}$, the degree of a vertex is the total affinity of its edges,

$$D_i = \sum_{j=1}^{N} w_{ij}, \tag{2.129}$$

Definition 2.5.4 *(Association between vertices).* Given $\mathbb{A}, \mathbb{B} \subset \mathbb{G}$, the relative association are calculated as follows,

$$\mathcal{A}(\mathbb{A}, \mathbb{B}) = \sum_{i\in\mathbb{A}} \sum_{j\in\mathbb{B}} w_{ij}, \tag{2.130}$$

which measures the total association between two subgraphs.

Definition 2.5.5 *(Volume).* Volume / size of a subgraph $\mathbb{A} \subseteq \mathbb{G}$ can be defined as follows,

$$vol(\mathbb{A}) = \sum_{i \in \mathbb{A}} d_i. \tag{2.131}$$

$$= \mathcal{A}(\mathbb{A}, \mathbb{A}). \tag{2.132}$$

Definition 2.5.6 *(Volume (alternate definition)).* Volume / size of a subgraph $\mathbb{A} \subseteq \mathbb{G}$ can also be alternately defined as the number of vertices as follows,

$$vol(\mathbb{A}) = |\mathbb{A}|. \tag{2.133}$$

Definition 2.5.7 *(Cut).* $\mathbb{G}$ can be partitioned into two disjoint subgraphs $\mathbb{A}$ and $\mathbb{B}$ where $\mathbb{A} \cup \mathbb{B} = \mathbb{G}$ and $\mathbb{A} \cap \mathbb{B} = \emptyset$, $\mathbb{B} = \mathbb{G} \backslash \mathbb{A}$ simply by removing connecting edges between them. The degree of dissimilarity between two subgraphs can be computed as total association of the edges that have been removed,

$$cut(\mathbb{A}, \mathbb{B}) = \mathcal{A}(\mathbb{A}, \mathbb{B}). \tag{2.134}$$

Definition 2.5.8 *(Minimum Cut).* A graph $\mathbb{G}$ can be cut such that the degree of affinity between $\mathbb{A}$ and $\mathbb{B}$ is minimized,

$$mincut(\mathbb{A}, \mathbb{B}) = \textbf{minimize } cut(\mathbb{A}, \mathbb{B}), \tag{2.135}$$

Definition 2.5.9 *(Ratio Cut [142]).* A graph $\mathbb{G}$ can be partitioned such that the degree of affinity between $\mathbb{A}$ and the other subgraphs $\mathbb{G} \backslash \mathbb{A}$ normalized with the number of vertices in each subgraph is minimized,

$$ratiocut(\mathbb{G}, K) = \textbf{minimize } \sum_{i}^{K} \frac{cut(\mathbb{A}_i, \mathbb{G} \backslash \mathbb{A}_i)}{|\mathbb{A}_i|}. \tag{2.136}$$

Definition 2.5.10 *(Normalized Cut [90]).* A graph $\mathbb{G}$ can be partitioned such that the degree of affinity between $\mathbb{A}$ and the other subgraphs $\mathbb{G} \backslash \mathbb{A}$ normalized with the weight of edges in each subgraph is minimized,

$$normalizedcut(\mathbb{G}, K) = \textbf{minimize } \sum_{i}^{K} \frac{cut(\mathbb{A}_i, \mathbb{G} \backslash \mathbb{A}_i)}{vol(\mathbb{A}_i)}. \tag{2.137}$$

2.5.1.2 Constructing Graphs

Several constructions that are often used to transform a given data points into a graph representation includes the *ε-neighborhood, k-nearest neighbor, and Gaussian neighborhood* [143].

Definition 2.5.11 *(ε-neighborhood graph).* Connect all vertices whose pairwise distance is less than ε such that,

$$w_{ij} = \begin{cases} 1 & if \quad ||\mathbf{v}_i - \mathbf{v}_j||^2 < \varepsilon \\ 0 & otherwise \end{cases}. \tag{2.138}$$

Definition 2.5.12 *(k-nearest neighbor graph).* Connect and assign the appropriate weight to the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ if either vertices are among the k-nearest neighbor of either $\mathbf{v}_i$ or $\mathbf{v}_j$,

$$w_{ij} = \begin{cases} a_{ij} & if \quad \mathbf{v}_i \in knn(\mathbf{v}_j) \quad or \quad \mathbf{v}_j \in knn(\mathbf{v}_i) \\ 0 & otherwise \end{cases}. \tag{2.139}$$

Definition 2.5.13 *(mutual k-nearest neighbor graph).* Connect and assign the appropriate weight to the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ if both vertices are among the k-nearest neighbor of $\mathbf{v}_i$ and $\mathbf{v}_j$,

$$w_{ij} = \begin{cases} w_{ij} & if \quad \mathbf{v}_i \in knn(\mathbf{v}_j) \quad and \quad \mathbf{v}_j \in knn(\mathbf{v}_i) \\ 0 & otherwise \end{cases}. \tag{2.140}$$

Definition 2.5.14 *(Gaussian neighborhood graph).* Local similarity is indicated by a pre-specified gaussian kernel such that,

$$a_{ij} = \phi(\mathbf{v}_i|\mathbf{v}_j, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{||\mathbf{v}_i - \mathbf{v}_j||^2}{2\sigma^2}\right). \tag{2.141}$$

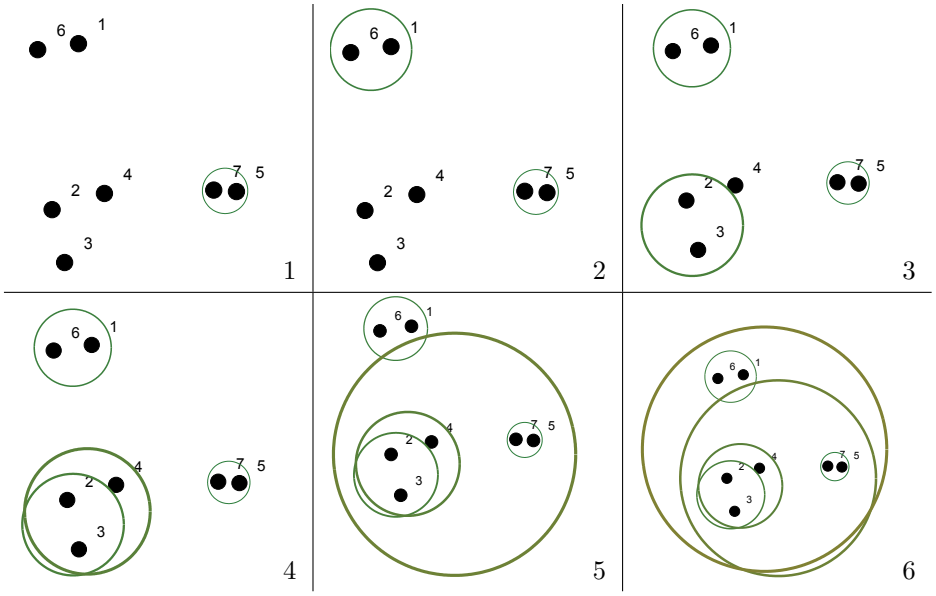
2.5.2 Agglomerative Linkage Clustering

Agglomerative linkage, or simply agglomerative clustering algorithms treat each object/vertex as a singleton cluster. Each cluster is recursively merged in pairs, until all clusters have been merged into a single cluster that contains all vertices. An illustration describing the process of agglomerative clustering can be seen in Figure 2.13.

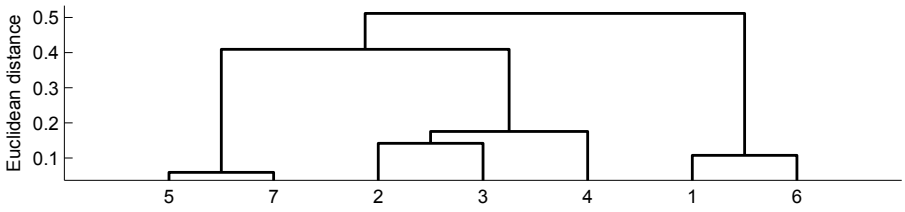
The updated proximity matrix can be calculated recursively using Lance-Williams [60] formula:

$$d(i+j,k) = \alpha_i d(i,k) + \alpha_j d(j,k) + \beta d(i,j) + \gamma |d(i,k) - d(j,k)|, \tag{2.142}$$

where $d(\cdot,\cdot)$ denotes the distance between two clusters, i, j, k denote the cluster indices. The Lance-Williams formula expresses the distance between a cluster k and a merged cluster $i+j$. The values of variables α_i, α_j, β, and γ are determined by the agglomeration criterion as outlined in Table 2.1.



(A) Recursive formation of clusters in agglomerative linkage clustering.



(B) Cluster dendrogram.

FIGURE 2.13: Agglomerative clustering using single linkage criteria.

TABLE 2.1: Agglomerative clustering: agglomeration criteria

Agglomeration Criterion	α_i	α_j	β	γ
Single Linkage	0.5	0.5	0	-0.5
Complete Linkage	0.5	0.5	0	0.5
Centroid Linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{n_i n_j}{(n_i+n_j)^2}$	0
Average Linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Median Linkage	0.5	0.5	-0.25	0
Ward Linkage	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$-\frac{n_k}{n_i+n_j+n_k}$	0

n_i, n_j, and n_k denote the cardinality of the set i, j and k, respectively.

From Table 2.1, it is easy to see that agglomeration using single linkage favors connected clusters while the others favors globular clusters. In fact, the single linkage algorithm closely resembles the minimum spanning tree (MST) algorithm [144]. The pseudocode can be seen in Algorithm 2.6.

Algorithm 2.6 Agglomerative clustering pseudocode

Input: $N \times N$ proximity matrix between each object / clusters.
Output: Cluster dendrogram.
1: **while** number of clusters > 1
2: Calculate α_i, α_j, β, and γ for each candidate (refer to Equation 2.142),
3: Merge two clusters with smallest linkage distance ($d(i+j,k)$),
4: Update the proximity matrix, reduce its order by 1.
5: **end while**

The agglomerative linkage is attractive and widely used due to its simplicity and capability of producing a tree-like cluster construction called dendrogram which can be useful in many fields including computational biology, document clustering, semantics, and genomic. Agglomerative linkage complexity is at least $\mathcal{O}(N^2)$.

2.5.3 Maximum Flow: Edmonds-Karp-Dinitz algorithm

This subsection is devoted to a polynomial time maximum-flow algorithm by Edmonds-Karp-Dinitz (1970-1972) [145, 146] which is based on Ford and Fulkerson's algorithm [147].

Definition 2.5.15 *(Maximum Flow).* Suppose we have a non-negative weighted directed graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$ where there exists a source node s and a sink node t, $(s,t) \in \mathbb{V}$. The admissible flow through an edge $e \in \mathbb{E}$ is constrained by its capacity $c(e)$. Every vertex $v \in \mathbb{V}\backslash\{s,t\}$, has to obey the *flow conservation constraint* such that the inflow $E^+(v)$ equals the outflow $E^-(v)$.

We wish to find the maximum flow $|f|(s \to t)$ such that,

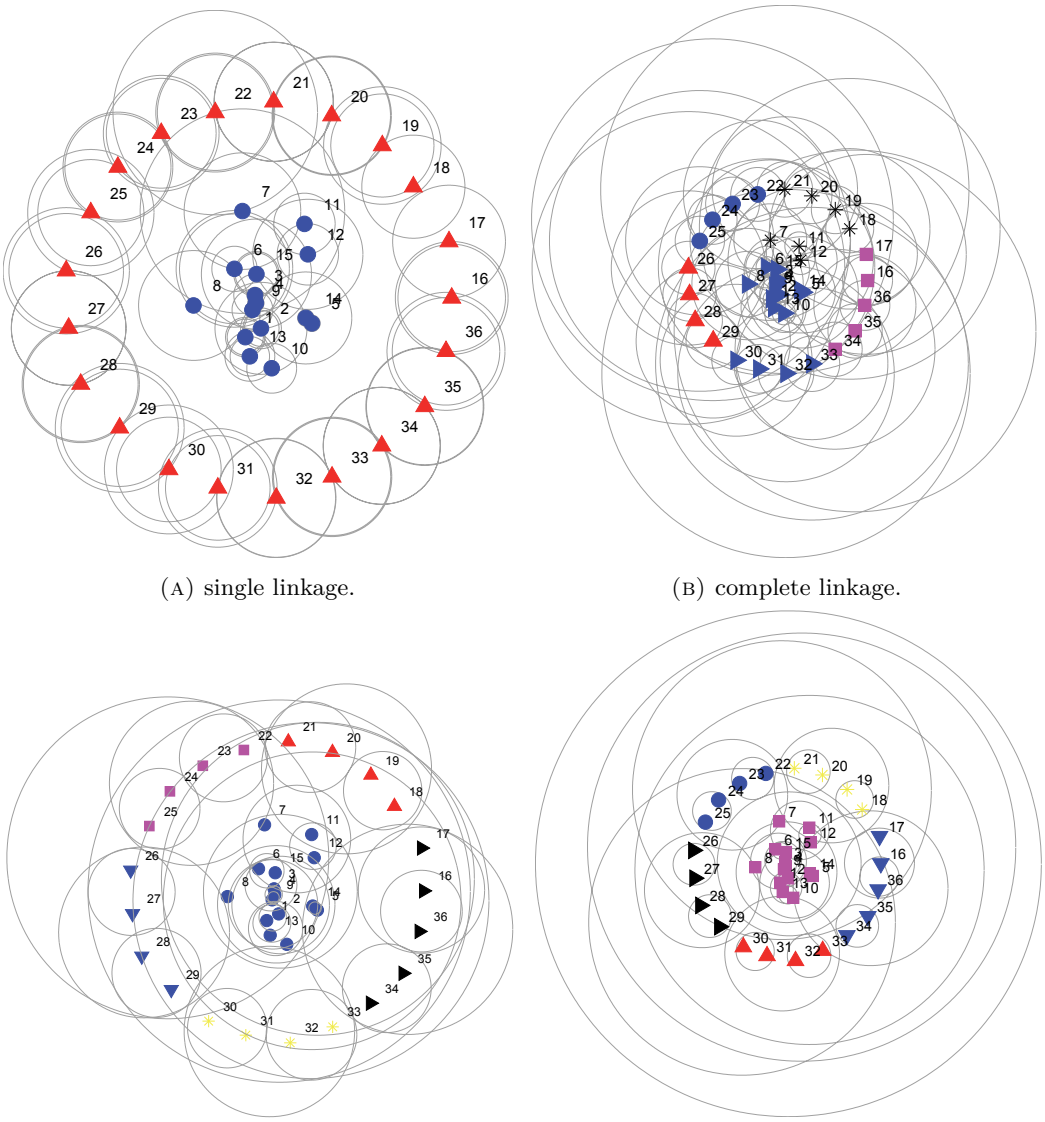
$$\textbf{Maximize} \quad |f| = \sum_{e \in E^-(s)} f(e), \tag{2.143}$$

$$\textbf{s.t.} \sum_{e \in E^-(v)} f(e) - \sum_{e \in E^+(v)} f(e) = 0, \qquad \forall v \in \mathbb{V}\backslash\{s,t\}, \tag{2.144}$$

$$0 \leq f(e) \leq c(e), \qquad e \in \mathbb{E}. \tag{2.145}$$

Definition 2.5.16 *($s-t$ cut).* An $s-t$ cut is a graph partition $(\mathbb{S}, \mathbb{V}\backslash\mathbb{S})$ into two disjoint subsets by removing edges $e \in \mathbb{C}$ such that $s \in \mathbb{S}$ and $t \in \mathbb{V}\backslash\mathbb{S}$. The capacity of $\mathbb{S}$ the cut is

$$c(\mathbb{S}, \mathbb{V}\backslash\mathbb{S}) = \sum_{e \in \mathbb{C}} c(e). \tag{2.146}$$



(A) single linkage.

(B) complete linkage.

(C) average linkage.

(D) ward linkage.

FIGURE 2.14: Illustration of agglomerative clustering using various linkage criteria

Definition 2.5.17 *(Flow Conservation Constraint).* The net flow out of any vertex v is equal to the net flow,

$$\sum_{e \in E^-(v)} f(e) - \sum_{e \in E^+(v)} f(e) = 0 \qquad \forall v \in V \backslash \{s, t\}. \tag{2.147}$$

Definition 2.5.18 *(Capacity Constraint).* $\forall e \in \mathbb{E}$, the maximum permissable flow is constrained by its capacity,

$$0 \leq f(e) \leq c(e), \qquad e \in \mathbb{E}. \tag{2.148}$$

Theorem 2.1 (Size of flow)**.** *The size of flow $|f|$ is the net outflow of the source vertex,*

$$|f| = \sum_{e \in E^-(s)} f(e). \tag{2.149}$$

Proof: As there is no inflow to the source node, by flow conservation constraint we have

$$|f| = \sum_{e \in E^-(s)} f(e) - \overbrace{\sum_{e \in E^+(s)} f(e)}^{=0}. \qquad \blacksquare \tag{2.150}$$

It follows that the net outflow of s is equal to the net inflow of t, hence $|f|$ may be interpreted as the amount of flow that is sent from s to t.

Definition 2.5.19 *(Residual capacity)***.** Given an edge $e \in \mathbb{E}$, its residual capacity $f_r(e)$ is its remaining capacity such that,

$$f_r(e) = c(e) - f(e). \tag{2.151}$$

Definition 2.5.20 *(Shortest augmenting path)***.** The *shortest augmenting path* $x_p \in \mathbb{P}$ is the shortest traversable path from s to t. Such path can be found in $\mathcal{O}(|\mathbb{E}|)$ time due to Dinitz [145] using a *breadth-first search* (BFS) algorithm on the *residual capacity* graph.

Definition 2.5.21 *(Bottleneck capacity)***.** Given a set of bottleneck edges $\mathbb{E}_b \subset \mathbb{E}$, a bottleneck capacity $c_b = c(e), (e \in \mathbb{E}_b)$ limits the maximum permissable $s \to t$ flow as constrained by its residual capacity.

Theorem 2.2 (Maximum-Flow/Minimum Cut Theorem)**.** *Let $|f|_{max}$ be a flow of maximum value and let cut$(\mathbb{S}, \mathbb{T})$ be the cut with minimum capacity, the minimum cut is achieved by removing edges with maximum capacity cut$(\mathbb{S}, \mathbb{T}) = |f|_{max}$.*

Proof: Let $\mathbb{E}_b \subset \mathbb{E}$ be the set of bottleneck edges and $\mathbb{P}$ be the shortest augmenting paths traversed to maximize $|f|$. Since the $|f|$ is maximal, the algorithm terminates as s is not anymore reachable from t due to the saturation of all bottleneck edges $(\forall e \in \mathbb{E}_b), f_r(e) = c(e) - f(e) = 0$. For each $x_p \in \mathbb{P}$, the outflow from s is constrained by its bottleneck capacity by definition. As we have exhausted all such path, the total outflow from s is then equal to the sum of all bottleneck capacities such that $|f| = \sum_{e \in \mathbb{E}_b} c(e) = c(\mathbb{S}, \mathbb{T}) = cut(\mathbb{S}, \mathbb{T})$. $\blacksquare$

Theorem 2.3 (Menger's Theorem)**.** *Let $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, a directed graph, and let $(s, t) \in \mathbb{V}$, the vertices in $\mathbb{G}$. The maximum weight among all flows from s to t in $\mathbb{D}$ equals the minimum capacity among all sets of edges $\mathbb{E}$ whose deletion destroys all directed paths from s to t [148].*

Proof: The proof follows directly from the Maximum-Flow/Minimum Cut theorem. $\blacksquare$

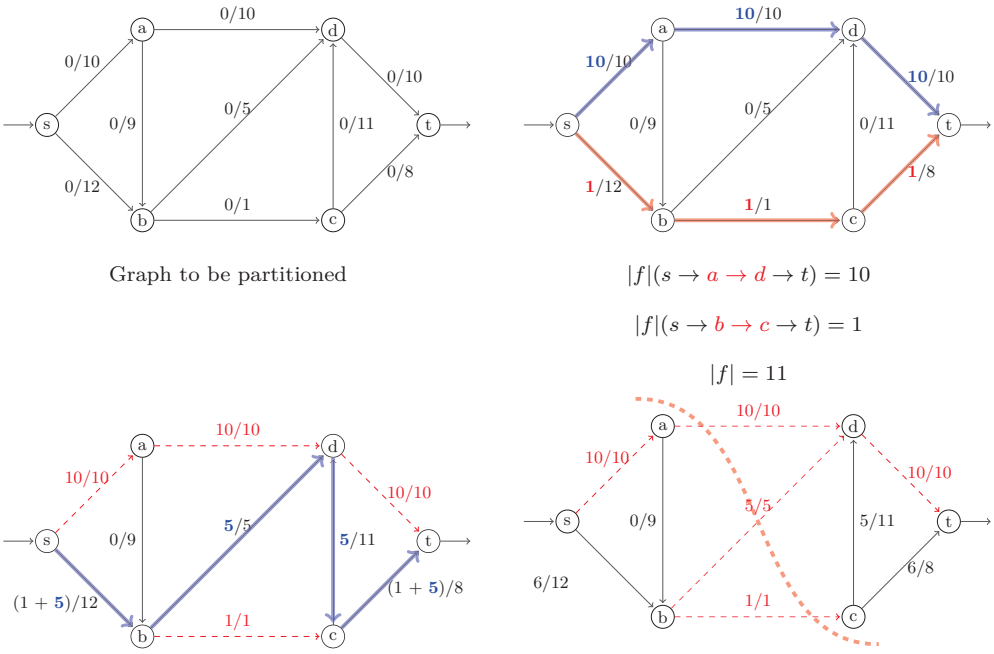
The EKD maximum flow algorithm pseudocode is provided in Algorithm 2.7.

Algorithm 2.7 Edmonds-Karp-Dinitz maximum flow pseudocode

Input: Graph $\mathbb{G} = \mathbb{V}, \mathbb{E}$, $s, t \in \mathbb{V}$.
Output: The maximum $s \to t$ flows $\mathbb{F}$.

1: $f(e) \xleftarrow{\forall e \in \mathbb{E}} 0$.
2: $c(e) \xleftarrow{\forall e \in \mathbb{E}} w(e)$.
3: Let the residual network $\mathbb{G}_r = \{\mathbb{V}, \mathbb{E}_r\}$, where $(\forall e \in \mathbb{E}), f_r(e) = f(e) - c(e)$
4: **while** There exists a permissable augmenting path $s \to t$ in $\mathbb{G}_r$
5: Let x_p be such path with the least number of edges.
6: Let $c_b = \arg\min_{e \in x_p} f_r(e)$, the bottleneck capacity of the path.
7: $f(e) \xleftarrow{\forall e \in x_p} f(e) + c_b$.
8: Recalculate $\mathbb{G}_r$.
9: **end while**
10: **return** $(\forall e \in \mathbb{E}), f(e)$.

Intuitively the algorithm terminates when t is not anymore reachable from s. The minimum cut are then indicated by the bottleneck edges which flows are at its maximum capacity. A simple illustration of the algorithm is shown in Figure 2.15. Edges in a this illustration is represented as flow/capacity (f/c) for clarity.



Graph to be partitioned

$|f|(s \to a \to d \to t) = 10$

$|f|(s \to b \to c \to t) = 1$

$|f| = 11$

$|f|(s \to b \to d \to c \to t) = 5$

$|f| = 11 + 5 = 16$

Maximum Flow = Minimum cut

FIGURE 2.15: A simple illustration of the Edmonds-Karp-Dinitz algorithm for finding the maximum flow / minimum cut of a graph. Bottleneck edges are noted in red in the equations.

2.5.3.1 Complexity Analysis

A drawback to the $s-t$ cut is that it returns a minimum cut given s and t, which is a locally minimum cut of $\mathbb{G}$, by definition. Consequently, the globally minimum cut can be computed in polynomial time by fixing s and varying t over all vertices, which implies $|\mathbb{V}|-1$ cut phases. The overall computational complexity of the algorithm is therefore $\mathcal{O}(|\mathbb{E}||\mathbb{V}|^2)$ due to the following. Calculation of BFS requires $\mathcal{O}(|\mathbb{E}|)$. As for each augmentation there is at least 1 edge saturated, the maximum flow at a given t can therefore be calculated in at most $\mathcal{O}(|\mathbb{E}||\mathbb{V}|)$ time. Reiterating over all possible t we have $\mathcal{O}(|\mathbb{E}||\mathbb{V}|^2)$.

2.5.4 Stoer-Wagner's Deterministic Minimum Cut Algorithm

Stoer-Wagner proposed a simple and elegant deterministic non-flow algorithm for calculating the global minimum cut of a graph [149]. The algorithm is devised to operate on undirected weighted graph $\mathbb{G}=\{\mathbb{V},\mathbb{E}\}$ with nonnegative real weights $(\forall e\in\mathbb{E}), w(e)$. Unlike flow algorithms, Stoer-Wagner's algorithm is insensitive to s and t initialization. To obtain the global optimum cut, one simply elect a starting node $a\in\mathbb{V}$, which can also be chosen randomly.

Theorem 2.4 (Stoer-Wagner's Theorem [149])**.** *Let s and t be two vertices of a graph $\mathbb{G}$. Let $\mathbb{G}/\{s,t\}$ be the graph obtained by merging s and t. A minimum cut of $\mathbb{G}$ can be obtained by taking the smaller of a minimum $s-t$-cut of $\mathbb{G}$ and a minimum cut of $\mathbb{G}/\{s,t\}$.*

The theorem holds since either there is a minimum cut of $\mathbb{G}$ that separates s and t, then a minimum $s-t$-cut of G is a minimum cut of G; or there is none, then a minimum cut lies elsewhere in the merged graph $\mathbb{G}/\{s,t\}$. This theorem implies that the globally minimum cut can be obtained by recursively finding arbitrary minimum $s-t$-cut of a graph on each $\mathbb{G}/\{s,t\}$ [149].

Definition 2.5.22 *(Contraction).* The contraction of an undirected graph $\mathbb{G}=\{\mathbb{V},\mathbb{E}\}$ by an edge $e(u,v)\in\mathbb{E},(u,v)\in\mathbb{V}$ is a graph $\mathbb{G}'=\{\mathbb{V}',\mathbb{E}'\}$ where $\mathbb{V}'=\mathbb{V}-\{u,v\}\cup\{x\}$. Here $x\in\mathbb{V}$ is a new vertex. The set of edges $\mathbb{E}'$ is formed from $\mathbb{E}$ by deleting the edge $e(u,v)$ and, for each vertex $w\in\mathbb{V}$ incident to u or v, deleting whichever of $e(u,w)\in\mathbb{E}$ and $e(v,w)\in\mathbb{E}$, and connecting it with the vertex x by adding the new edge $e(x,w)$. The weight of the consolidated edges is the sum of the two edges that now coincides, e.g. for the deletion of $e(v,w)$ we have $w_{\{u,w\},x}=w_{u,x}+w_{w,x}$.

The time complexity of contraction is $\mathcal{O}(|\mathbb{V}|)$.

Definition 2.5.23 *(Most tightly connected vertex).* Let $\mathbb{A}\subset\mathbb{V}$ be a subgraph. The most tightly connected vertex $z\in\mathbb{V}$ is the vertex $y\in\mathbb{V}, y\notin\mathbb{A}$ incident to $\mathbb{A}$ which association $\mathcal{A}(\mathbb{A},y)$ is

maximum,

$$z = \arg\max_{y \notin \mathbb{A}} \mathcal{A}(\mathbb{A}, y). \tag{2.152}$$

Recall that the association between $\mathbb{A}$ to y refers to total weights of the connections from $\mathbb{A}$ to the vertex y.

Stoer-Wagner's algorithm is as follows. Two most tightly connected edges are recursively contracted starting from an arbitrary vertex $a \in \mathbb{V}$. The algorithm pseudocode is shown in Algorithm 2.8.

Algorithm 2.8 Stoer-Wagner minimum cut pseudocode

Input: Weighted graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, edge weights $(\forall e \in \mathbb{E}), w(e)$, and any starting vertex $a \in \mathbb{V}$.
Output: The global minimum cut $\mathbb{MC} = \{\{\mathbb{S}\}, \{\mathbb{V} \backslash \mathbb{S}\}\}$.

$\mathbb{A} \leftarrow \{a\}$.
while $|\mathbb{V}| > 1$
  $\{\mathbb{CP}, \mathbb{G}\} \leftarrow MinimumCutPhase(\mathbb{G}, w, a)$
  if $cut(\mathbb{CP}) < cut(\mathbb{MC})$
    $\mathbb{MC} \leftarrow \mathbb{CP}$
  end if
end while
return $\mathbb{MC}$.

Input: Weighted graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, any starting vertex $a \in \mathbb{V}$.
Output: min-cut of the phase $\mathbb{CP} = \{\{\mathbb{A}\}, \{\mathbb{V} \backslash \mathbb{A}\}\}$, contracted graph $\mathbb{G}' = \{\mathbb{V}', \mathbb{E}'\}$.

function $MinimumCutPhase(\mathbb{G}, w, a)$.
$\mathbb{A} \leftarrow \{a\}$.
while $\mathbb{A} \neq \mathbb{V}$
  let $z \notin \mathbb{A}$ be the most tightly connected vertex to $\mathbb{A}$.
  if $|\mathbb{A}| - |\mathbb{V}| == 2$
    $s \leftarrow z$.
  end if
  if $|\mathbb{A}| - |\mathbb{V}| == 1$
    $t \leftarrow z$.
    $\mathbb{CP} \leftarrow \{\{\mathbb{A}\}, \{\mathbb{V} \backslash \mathbb{A}\}\}$.
    $\mathbb{G}' \leftarrow contract((s, t) \in \mathbb{G})$
  end if
  $\mathbb{A} \leftarrow contract(\mathbb{A}, z)$.
end while
return $\{\mathbb{CP}, \mathbb{G}'\}$.
end function

2.5.4.1 Complexity Analysis

Stoer and Wagner proposes that the overall complexity of the algorithm is $\mathcal{O}(|E||V| + |V|^2 \log |V|)$ [149] due to the following. The complexity of the algorithm lies in the *MinimumCutPhase* function. The computation for contract can be made efficient using priority queue and Fibonacci heaps. During each phase we assign every vertices not in $\mathbb{A}$ in the priority queue. On each contraction, we need to remove the most tightly connected vertex by removing it from the queue. For each deletion HEAP-EXTRACTMAX ($\mathcal{O}(\log |\mathbb{V}|)$) need to be called once. The weight for the consolidated edge can

be calculated easily by calling HEAP-INCREASEKEY ($\mathcal{O}(1)$). Hence *MinimumCutPhase* requires overall $|\mathbb{V}|$ HEAP-EXTRACTMAX and $|\mathbb{E}|$ HEAP-INCREASEKEY operations, which accumulates to $\mathcal{O}(|\mathbb{E}| + |\mathbb{V}| \log |\mathbb{V}|)$. The total complexity for investigating all vertices in $\mathbb{V}$ is therefore $\mathcal{O}(|\mathbb{E}||\mathbb{V}| + |\mathbb{V}|^2 \log |\mathbb{V}|)$ [149]. An illustration of the execution of the Stoer-Wagner's algorithm on a simple graph can be seen in Figure 2.16.

2.5.4.2 Proof of correctness

Lemma 2.5.1 *(Optimality of MinimumCutPhase).* ***.*** *MinimumCutPhase would always return a minimum $s-t$ cut for $\mathbb{G}$.*

Proof: The proof by induction below is quoted from Stoer-Wagner's manuscript [149].

Let $\mathbb{C} = (\mathbb{S}, \mathbb{V} \backslash \mathbb{S})$ be an arbitrary $s-t$ cut and let $\mathbb{CP}$ be the cut of the phase. Let $(u, v) \in \mathbb{V}$ be the two vertices added during an execution of *MinimumCutPhase*, where v is added to $\mathbb{A}$ before u.

We define two sets $\mathbb{A}_u$ and $\mathbb{A}_v$ as two sets of vertices where $\mathbb{A}_u$ is a set of vertices added to $\mathbb{A}$ before u and $\mathbb{A}_v$ is similarly defined. Let $\mathbb{C}_u$ be the induced cut of the set $\mathbb{A}_u \cup u$, and $\mathbb{C}_v$ be the induced cut of the set $\mathbb{A}_v \cup v$. Clearly when $s \leftarrow u$ and $t \leftarrow v$, then $\mathbb{C}_v = \mathbb{CP}$.

We define $w(\mathbb{A}_u, u)$ as the sum of weights in $\mathbb{A}_u$ connected to u, and $w(\mathbb{C}_u)$ as the sum of weights in $\mathbb{C}_u$ after inducing the cut. It is then obvious that $\forall u, w(\mathbb{A}_u, u) \leq w(\mathbb{C}_u)$ by definition since the edge connecting $\mathbb{A}_u$ to u is the only edges that cross the cut at $\mathbb{C}_u$.

The proof can be provided by induction,

$$\begin{aligned} w(\mathbb{A}_u, u) &= w(\mathbb{A}_v, u) + w(\mathbb{A}_u \backslash \mathbb{A}_v, u) \\ &\leq w(\mathbb{C}_v) + w(\mathbb{A}_u \backslash \mathbb{A}_v, u) \\ &\leq w(\mathbb{C}_u). \end{aligned} \tag{2.153}$$

Since t is always an active vertex due to the fact that it is merged last, we can conclude that:

$$\begin{aligned} w(\mathbb{A}_t, t) &\leq w(\mathbb{C}_t) \\ &\leq w(\mathbb{C}), \end{aligned} \tag{2.154}$$

which states directly that the $\mathbb{CP}$ would be at most as heavy as $\mathbb{C}$. ■

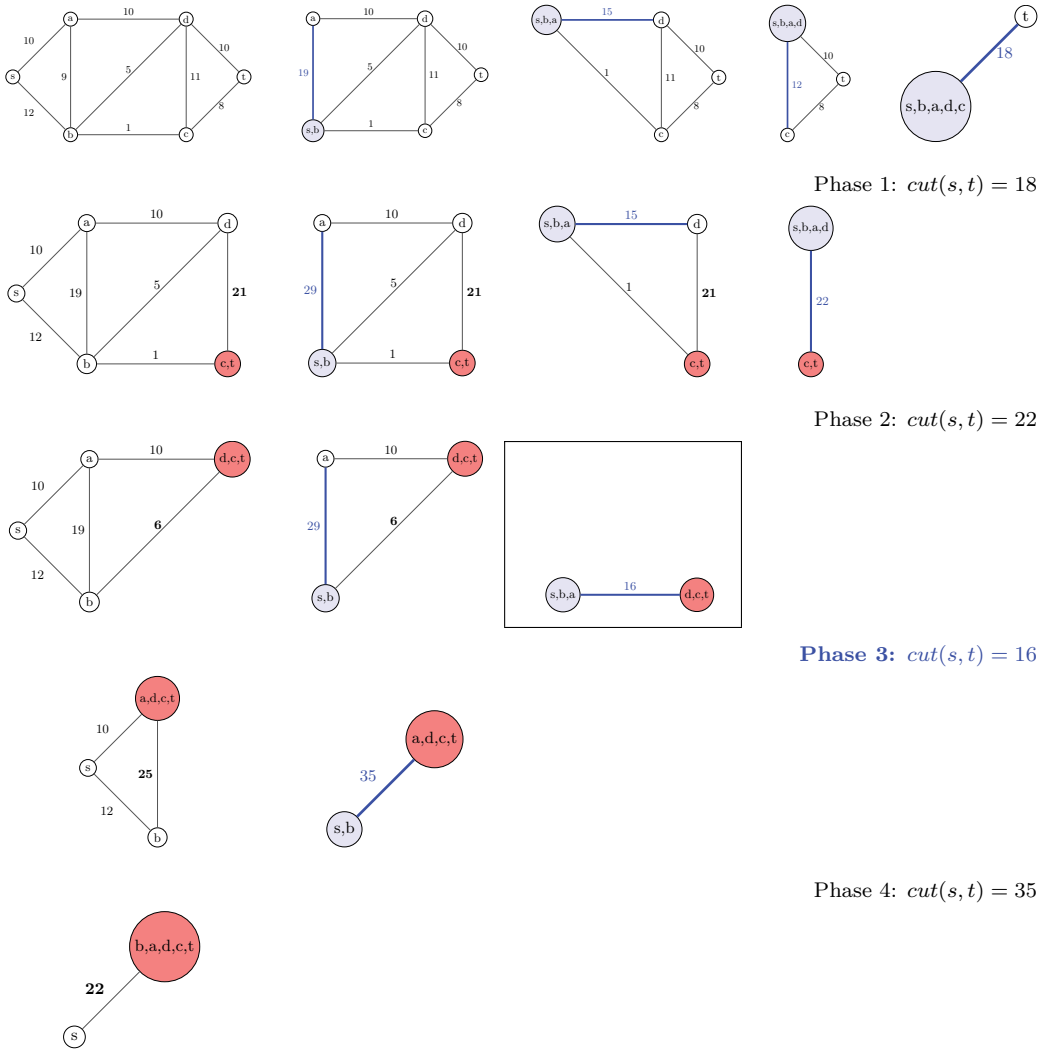


FIGURE 2.16: A simple illustration of Stoer-Wagner's algorithm. The minimum cut is obtained at Phase 3 ({s,b,a}{d,c,t}).

2.5.5 Stochastic Flows

Stochastic flow simulates random walk through the connected vertices in the graph [150]. The weight of each edge represents the strength of interaction between two vertices. Intuitively, the more adjacent two vertices are, the stronger the interaction and therefore the more probable a random walker would traverse through the edge. In other words, in stochastic flows, adjacency matrix functions as the Markov probability transition matrix.

2.5.5.1 Theoretical Framework

Definition 2.5.24 *(Stochastic Matrix).* Given an undirected weighted graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$ with nonnegative real weight $(\forall e \in \mathbb{E}), w(e)$ stored in an adjacency matrix $\mathbf{A}_{[|\mathbb{V}|\times|\mathbb{V}|]}$. We can calculate $\mathbf{M}_{[|\mathbb{V}|\times|\mathbb{V}|]}$, a column-stochastic matrix, where $(\forall j), \sum_{i=1}^{|V|} m_{ij} = 1$, such that the i^{th} column of $\mathbf{M}$ represents the probability of transitioning from vertex v_i to v_j, $(v_i, v_j \in \mathbb{V})$ as follows,

$$\mathbf{M} = \mathbf{A}\mathbf{D}^{-1}, \tag{2.155}$$

where $\mathbf{D}$ is the diagonal degree matrix with $d_{ii} = \sum_{j=1}^{|\mathbb{V}|} a_{ij}$. The matrix $\mathbf{M}$ is also referred to as the canonical transition matrix $\mathbf{M_G}$.

Theorem 2.5 (Stationary Distribution)**.** *A stationary probability vector π^T is a row vector whose values do not change under application of the transition matrix,*

$$\pi^T\mathbf{M} = \pi^T. \tag{2.156}$$

Theorem 2.6 (Perron-Frobenius Theorem)**.** *A random walk corresponds to a homogenous Markov chains with transition probability of* $\mathbf{M}$*. This matrix is ergodic and will converge to its stationary distribution π independent of any initial distribution i,*

$$\lim_{r\to\infty} (\mathbf{M}^r)_{ij} = \pi_j^T, \tag{2.157}$$

where π_j is the j^{th} row of the matrix π. According to Perron-Frobenius theorem on stochastic matrix, the largest eigenvalue of such matrix is always equal to 1 ($\lambda_1 = 1$). The mixing speed or convergence rate will be determined by the conditions of the other eigenvectors where $\lambda_2 \geq \lambda_3 \geq \ldots \lambda_{|\mathbb{V}|}$.

Proof: Let us define an eigensystem $\mathbf{x}^T\mathbf{M} = \lambda\mathbf{x}$. Since $\mathbf{M}$ is a column-stochastic matrix, by definition we have $0 \leq m_{ij} \leq 1$ and $\sum_i m_{ij} = 1$ which makes $\mathbf{x}^T\mathbf{M}$ a convex matrix multiplication for any value < 1. In fact, the Perron-Frobenius eigenvector for this eigensystem is a vector of ones $\mathbf{1}$. The eigenvalue of such vector is then to be expected: $\mathbf{1}^T\mathbf{M} = 1$. Since $\overbrace{\lambda_1}^{=1} \geq \lambda_2 \geq \lambda_3 \geq \ldots \lambda_{|\mathbb{V}|}$, with sufficiently large r the eigenvalues will vanish at a specific rate such that $\overbrace{\lambda_1^r}^{=1} \geq \overbrace{\lambda_2^r \geq \lambda_3^r \geq \ldots \lambda_{|\mathbb{V}|}^r}^{<<1}$ as $r \to \infty$. This theorem is an extensively large subject in Markovian literature [151] hence the proof presented here, although incomplete, is sufficient to show the

correctness of the theorem. Readers are encouraged to refer to [151] for in-depth discussion and proofs on mixing rate and uniqueness of such π. ■

Theorem 2.7 (Markov Chain and its Spectral Connection)**.** *Given there is a clear separation between clusters, for large r, cluster assignments are shown in the eigenvectors.*

Proof: We can express our Markov chain as follows [152],

$$\begin{aligned}
\mathbf{P}(t) &= \mathbf{A}\mathbf{D}^{-1}\mathbf{P}(t-1) \\
\mathbf{P}(t) &= \mathbf{D}^{1/2}\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]\mathbf{D}^{-1/2}\mathbf{P}(t-1) \\
&= \mathbf{D}^{1/2}\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]\mathbf{D}^{-1/2}\mathbf{P}(t-2) \\
&= \mathbf{D}^{1/2}\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]^{r}\mathbf{D}^{-1/2}\mathbf{P}(t-r).
\end{aligned} \tag{2.158}$$

As of Equation 2.158, the ergodicity of the Markov Chains clearly expresses a rather interesting spectral connection as follows,

$$\mathbf{M}^{r} = \mathbf{D}^{1/2}\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]^{r}\mathbf{D}^{-1/2}. \tag{2.159}$$

The symmetric matrix $\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]$ can be written in term of its eigendecomposition,

$$\left[\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right]^{r} = \lambda_1^r\mathbf{x}_1\mathbf{x}_1^T + \lambda_2^r\mathbf{x}_2\mathbf{x}_2^T + \ldots + \lambda_{|\mathbb{V}|}^T\mathbf{x}_{|\mathbb{V}|}\mathbf{x}_{|\mathbb{V}|}^T. \tag{2.160}$$

The eigenvalues for this symmetric matrix are obviously the same as those of $\mathbf{P}$, while the eigenvectors are premultiplied with $\mathbf{D}^{-1/2}$. Since the largest eigenvalue $\lambda_1 = 1$, and $\lambda_{\{2,\ldots,|\mathbb{V}|\}} < 1$, as $r \to \infty$ we have,

$$\mathbf{M}^{\infty} = \mathbf{D}^{-1/2}\mathbf{x}_1\mathbf{x}_1^T\mathbf{D}^{-1/2}, \tag{2.161}$$

if and only if $\lambda_2, \lambda_3, \ldots, \lambda_{|\mathbb{V}|}$ vanishes at $m \to \infty$.

Consider now that a graph contains two subgraphs $(\mathbb{A}, \mathbb{B}) \subset \mathbb{G}, \mathbb{A} = \mathbb{G}\backslash\mathbb{B}$. In such situation there is a relatively small probability to transition from vertices in $\mathbb{A}$ to $\mathbb{B}$ and vice versa. With $\lambda_1 = 1$, $\lambda_2 < 1$, at least the second eigenvector shall remain strong for sufficiently large r, hence we have,

$$\mathbf{M}^{r} \approx \mathbf{D}^{-1/2}\left(\mathbf{x}_1\mathbf{x}_1^T + \lambda_2^r\mathbf{x}_2\mathbf{x}_2^T\right)\mathbf{D}^{-1/2}. \tag{2.162}$$

Looking at the transition matrix produced by the second eigenvector $(\mathbf{x}_2\mathbf{x}_2^T)_{ij} = \mathbf{x}_{2i}\mathbf{x}_{2j}^T$ we understand that these are the indicators for the subgraphs $\mathbb{A}$ and $\mathbb{B}$ since the resulting matrix would have a strong transition between i and j when both have the same sign. It then follows easily that for K clusters, there will be at least K nonzero eigenvalues. ■

We now have the sufficient theoretical framework to proceed with the formal clustering algorithm.

2.5.5.2 Markov Clustering Algorithm

Stochastic flow is also known as the Markov Clustering Algorithm (MCL), originally proposed by van Dongen in 2000 [150]. In 2009, Satuluri and Parthasarathy proposed the Regularized MCL (R-MCL), introducing a regularization parameter for preventing oversegmentation [153]. Satuluri and Parthasarathy also appropriately addresses the scalability issue apparent in MCL. In the manuscript a multiscale R-MCL using graph coarsening is proposed.

The MCL algorithm is an iterative process of applying two operators, namely expansion and inflation, on an initial stochastic matrix $\mathbf{M}$, in alternation, until convergence where $\mathbf{M}$ expresses a single connection to the central node of each cluster (e.g. only 1 nonzero element in each column). At convergence, eigendecomposition of $\mathbf{M}$ will reveal K eigenvectors with nonzero eigenvalues where K is the number of cluster, in conjunction with Theorem 2.7.

Definition 2.5.25 *(Expansion).* Expansion can be understood as a random walk, starting from an initial distribution $\mathbf{M}$, hence strengthening the first eigenvector and weakening the rest. The operation is as follows,

$$\mathbf{M}_{expansion} = Expand(\mathbf{M}) = \mathbf{MM}. \tag{2.163}$$

Definition 2.5.26 *(Inflation).* Inflation can be understood as a *graph thickening* operation. This operation *strengthens* stronger flows and *weakens* already weak flows. The operation is as follows,

$$\mathbf{M}_{inflation} = Inflate(\mathbf{M}) = \langle \mathbf{M} \rangle^t \mathbf{D}^{-1}_{\langle \mathbf{M} \rangle^t}, \tag{2.164}$$

$$\mathbf{D}_{\langle \mathbf{M} \rangle^t} = (\sum_j \langle \mathbf{M} \rangle^t), \tag{2.165}$$

where $\langle \cdot \rangle^t$ denotes t^{th} Hadamard power (element-wise power) of a matrix. $\mathbf{D}^{-1}_{\langle \mathbf{M} \rangle^t}$ corresponds to the inverse of the (diagonal) degree matrix of $\langle \mathbf{M} \rangle^t$, which normalizes each column such that it sums to 1.

Definition 2.5.27 *(Prune).* In each column, values that are 'small' in relation to the other entries are pruned to conserve memory. The heuristics utilized in the pruning operation employs a dynamic threshold based on the mean and maximum value of a column as follows,

$$th_j = \alpha \frac{1}{|\mathbb{V}|} \sum_j \mathbf{M}_j^2 \left(1 - \beta \left[max_j(\mathbf{M}_j^2) - \frac{1}{|\mathbb{V}|} \sum_j \mathbf{M}_j^2 \right] \right), \tag{2.166}$$

where $0 < \alpha \leq 1$ and $\beta = \{1, 8\} \in \mathbb{R}$. The motivation of the heuristic depends on the fact that if the maximum value of a column is close to its mean, then the probability distribution of the large nonzero component in that column is rather uniformly distributed [150].

Another possible simpler heuristic is by only considering the mean as follows,

$$th_j = \alpha \left(\frac{1}{|\mathbb{V}|} \sum_j \mathbf{M}_j^2 \right)^{\beta}, \tag{2.167}$$

where $0 < \alpha \leq 1 \leq \beta$. Finally the prune operation is carried as follows,

$$Prune(\mathbf{M}) = \mathbf{M} \xleftarrow{(\forall i, \forall j)} m_{ij} = \begin{cases} m_{ij} & if \quad m_{ij} > th_j \\ 0 & otherwise \end{cases} \tag{2.168}$$

Definition 2.5.28 *(Interpretation).* At convergence, $\mathbf{M}$ will express a single connection to the central node such that each column has only 1 nonzero element. These central nodes should be interpreted as the cluster centroids which govern the surrounding data points. One can perform an Eigendecomposition on $\mathbf{M}$, which would reveal K eigenvectors with nonzero eigenvalues where K is the number of clusters as a consequence of Theorem 2.7.

The pseudocode of MCL is shown in Algorithm 2.9.

Algorithm 2.9 Markov Clustering (MCL) algorithm pseudocode

Input: Adjacency matrix $\mathbf{A}$, inflation constant t, pruning constant α, β.
Output: Crisp clustering U.

1: $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{I}$
2: $\mathbf{M} \leftarrow \mathbf{A}\mathbf{D}^{-1}$
3: **repeat**
4: $\quad \mathbf{M} \leftarrow Expand(\mathbf{M})$
5: $\quad \mathbf{M} \leftarrow Inflate(\mathbf{M}, t)$
6: $\quad \mathbf{M} \leftarrow Prune(\mathbf{M}, \alpha, \beta)$
7: **until** $\mathbf{M}$ converges,
8: **return** $\mathsf{U} \leftarrow interpret(\mathbf{M})$

2.5.5.3 Regularized Markov Clustering

With motivation to normalizes the edges in a graph with respect to their co-connectivity, Satuluri and Parthasarathy [153] suggest normalizing $\mathbf{A}$ as follows,

$$\mathbf{A} \xleftarrow{(\forall i, \forall j)} a_{ij} = \frac{a_{ij}}{d_{ii}} + \frac{a_{ij}}{d_{jj}}, \tag{2.169}$$

where d_{ii} and d_{jj} are the degree of vertices i and j respectively.

Lemma 2.5.2 *(Expansion Regularization).* *Let* $\mathbf{q}_i$ *be the current flow distribution of a vertex in a graph and* w_i *be the respective normalized weight,* $\sum_i w_i = 1$. *The distribution* $\mathbf{q}^*$ *for each vertex in* $\mathbf{q}_i$ *minimizes its divergence with respect to its neighbors when*

$$\mathbf{q}^*(x) = \sum_{i=1}^{|\mathbb{V}|} w_i \mathbf{q}_i(x). \tag{2.170}$$

Proof: We wish to find the distribution $\mathbf{q}^*$ for each vertex such that its divergence with respect to its neighbors is minimized. Formally, we can express this regularization requirement in terms of minimizing the KL-divergence between each column [153],

$$\begin{aligned} \underset{\mathbf{q}^*(x)}{\text{minimize}} \quad & f(\mathbf{q}^*(x), \mathbf{q}_i(x)) \sum_{i=1}^{|\mathbb{V}|} w_i KL(\mathbf{q}_i(x) || \mathbf{q}^*(x)), \\ \text{s.t.} & \sum_x \mathbf{q}^*(x) = 1, \end{aligned} \tag{2.171}$$

where we know that $KL(\mathbf{q}_i(x)||\mathbf{q}^*(x)) = \sum_x (-\mathbf{q}_i(x) \log \mathbf{q}^*(x) + \mathbf{q}_i(x) \log \mathbf{q}_i(x))$. Using lagrange multiplier λ for the constraint and taking the first derivative w.r.t $\mathbf{q}^*(x)$ equals to 0 we have,

$$\begin{aligned} L(\mathbf{q}^*(x), \mathbf{q}_i(x)) &= \sum_{i=1}^{|\mathbb{V}|} w_i \left(\sum_x \left[-\mathbf{q}_i(x) \log \mathbf{q}^*(x) + \mathbf{q}_i(x) \log \mathbf{q}_i(x) \right] \right) + \lambda \left(\sum_x \mathbf{q}^*(x) - 1 \right) \\ &= \sum_{i=1}^{|\mathbb{V}|} \sum_x w_i \left[-\mathbf{q}_i(x) \log \mathbf{q}^*(x) + \mathbf{q}_i(x) \log \mathbf{q}_i(x) \right] + \lambda \left(\sum_x \mathbf{q}^*(x) - 1 \right) \\ \frac{\partial L(\mathbf{q}^*(x), \mathbf{q}_i(x))}{\partial \mathbf{q}^*(x)} &= -\sum_{i=1}^{|\mathbb{V}|} w_i \frac{\mathbf{q}_i(x)}{\mathbf{q}^*(x)} + \lambda = 0 \\ \mathbf{q}^*(x) &= \sum_{i=1}^{|\mathbb{V}|} \frac{w_i \mathbf{q}_i(x)}{\lambda}. \end{aligned} \tag{2.172}$$

Imposing the constraint $\sum_x \mathbf{q}^*(x) = 1$,

$$\begin{aligned} \sum_x \mathbf{q}^*(x) &= \sum_x \sum_{i=1}^{|\mathbb{V}|} \frac{w_i \mathbf{q}_i(x)}{\lambda} = 1 \\ \lambda &= \sum_{i=1}^{|\mathbb{V}|} w_i \overbrace{\sum_x \mathbf{q}_i(x)}^{=1} = \overbrace{\sum_{i=1}^{|\mathbb{V}|} w_i}^{=1} = 1. \end{aligned} \tag{2.173}$$

Finally, substituting λ into Equation 2.172 we have a minimum at $\mathbf{q}^*(x) = \sum_{i=1}^{|\mathbb{V}|} w_i \mathbf{q}_i(x)$. ■

A consequence of Lemma 2.5.2 is Satuluri and Parthasarathy's redefinition of expansion [153] as given in the following.

Corollary 2.5.1 *(Regularized Expansion). Due to Lemma 2.5.2, it follows that the integrity of the stochastic matrix* $\mathbf{M}$ *can be conserved by regularizing its expansion on its canonical distribution* $\mathbf{M_G}$*. Satuluri and Parthasarathy referred this proposition in their original manuscript as* 'regularize' *[153] which mechanism is as follows.*

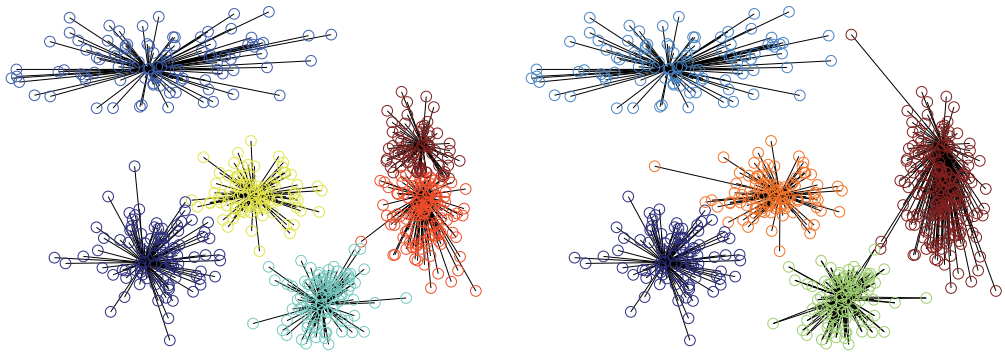
$$\mathbf{M}_{regularized} = Regularize(\mathbf{M}, \mathbf{M_G}) = \mathbf{M}\mathbf{M_G}. \tag{2.174}$$

The algorithmic proposition of R-MCL is thus shown in Algorithm 2.10. An illustration of the final clustering output for both MCL and R-MCL is shown in Figure 2.17.

Algorithm 2.10 Regularized Markov Clustering (R-MCL) algorithm pseudocode

Input: Adjacency matrix $\mathbf{A}$, inflation constant t, pruning constant α, β.
Output: Crisp clustering U.

1: $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{I}$,
2: $\mathbf{M} \leftarrow \mathbf{M_G} \leftarrow \mathbf{A}\mathbf{D}^{-1}$.
3: **repeat**
4: $\quad \mathbf{M} \leftarrow Regularize(\mathbf{M}, \mathbf{M_G})$,
5: $\quad \mathbf{M} \leftarrow Inflate(\mathbf{M}, t)$,
6: $\quad \mathbf{M} \leftarrow Prune(\mathbf{M}, \alpha, \beta)$,
7: **until** $\mathbf{M}$ converges,
8: **return** $\mathsf{U} \leftarrow interpret(\mathbf{M})$.



(A) MCL (B) R-MCL

FIGURE 2.17: Illustration of MCL and R-MCL for clustering the Five Gaussians dataset

2.5.6 Spectral Clustering

The spectral clustering emerges as a relatively novel paradigm to data partitioning based on the spectral graph theory [61, 90]. It is relatively simple to implement and can be solved efficiently

using standard linear algebra.

The minimum cut of a graph can be obtained using various algorithms from the Maximum Flow family or Stoer-Wagner deterministic cut as has been covered in the previous subsections. Spectral clustering, however, addresses graph partitioning from a different point of view. As a general remark, the spectral clustering concerns more on the analysis of a graph in term of its spectral characteristics in the eigenspace. Such paradigm allows spectral clustering to recover roughly the automatic number of cluster based on its eigendecomposition. However a well known problem with spectral clustering is its oversensitivity to noisy information.

2.5.6.1 Ratio Cut

Definition 2.5.29 *((Unnormalized) Graph Laplacian).* Given an weighted undirected graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$ with the adjacency / weight between vertices stored in $\mathbf{A}$. The unnormalized graph Laplacian matrix is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.175}$$

Theorem 2.8 (Unnormalized Graph Laplacian and Ratio Cut [142]). *For every vector* $\mathbf{u} \in \mathbb{R}^{|\mathbb{V}|}$, $\mathbf{L}$ *satisfies the following property,*

$$\mathbf{u}^T\mathbf{L}\mathbf{u} = \frac{1}{2}\sum_i\sum_j a_{ij}(u_i - u_j)^2, \tag{2.176}$$

which minimization yields the ratio cut of $\mathbb{G}$.

Proof: by definition of the degree of a vertex $d_i = \sum_j a_{ij}$, $d_j = \sum_i a_{ij}$, and $d_i = d_j$ we have,

$$\begin{aligned}
\mathbf{u}^T\mathbf{L}\mathbf{u} &= \mathbf{u}^T\mathbf{D}\mathbf{u} - \mathbf{u}^T\mathbf{A}\mathbf{u}, \\
&= \sum_i d_i u_i^2 - \sum_i\sum_j a_{ij}u_i u_j, \\
&= \frac{1}{2}\left(\sum_i\sum_j a_{ij}u_i^2 - 2\sum_i\sum_j a_{ij}u_i u_j + \sum_i\sum_j a_{ij}u_j^2\right), \\
&= \frac{1}{2}\sum_i\sum_j a_{ij}(u_i - u_j)^2. \qquad \square
\end{aligned} \tag{2.177}$$

Recall the objective function of ratio cut,

$$ratiocut(\mathbb{G}, K) = \text{minimize } \sum_i^K \frac{cut(\mathbb{A}_i, \mathbb{G}\backslash\mathbb{A}_i)}{|\mathbb{A}_i|},$$

where $\mathbb{A}_i$ denotes the i^{th} subgraph. If we define a cluster assignment matrix $\mathbf{U}_{N\times K}$, which we relax such that it contains real numbers with the constraint $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, we can express the ratio cut as a trace minimization problem,

$$\begin{aligned} ratiocut(\mathbb{G}, K) = &\underset{\mathbf{U}}{\text{minimize}} \quad Tr\left(\mathbf{U}^T\mathbf{L}\mathbf{U}\right). \\ &\underset{\mathbf{U}}{\text{minimize}} \quad \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{|\mathbb{V}|}\sum_{j=1}^{|\mathbb{V}|} a_{ij}(u_{ik} - u_{jk})^2. \\ &\text{s.t.} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{aligned} \tag{2.178}$$

Using Lagrange multiplier λ for the constraint, the solution to the minimization problem can be obtained by taking the first partial derivative of the Lagrangian function with respect to $\mathbf{U}$ and setting it to 0 as follows.

$$\begin{aligned} L(\mathbf{U}) &= \mathbf{U}^T\mathbf{L}\mathbf{U} - \lambda\left(\mathbf{U}^T\mathbf{U} - \mathbf{I}\right), \\ \frac{\partial L}{\partial \mathbf{U}} &= 2\mathbf{L}\mathbf{U} - 2\lambda\mathbf{U}, \\ &= (\mathbf{L} - \lambda\mathbf{I})\,\mathbf{U} = 0. \end{aligned} \tag{2.179}$$

Equation 2.179 is readily recognizable as an eigensystem formulation for $\mathbf{L}$. The first eigenvector provides the minimum eigenvalue of 0 ($\lambda_1 = 0$) however it does not carry the necessary information since all entries in $\mathbf{u}_1$ are $1/\sqrt{|\mathbb{V}|}$. The ratio cut solution is obtained by observing the first K eigenvectors starting from the second, $\mathbf{u}_2, \mathbf{u}_3, \ldots, \mathbf{u}_K, \ldots$ where $0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_K \leq \ldots \leq \lambda_{|\mathbb{V}|}$ [142]. ■

The ratio cut algorithm pseudocode can be seen in Algorithm 2.11.

Algorithm 2.11 Spectral clustering (ratio cut) algorithm pseudocode

Input: Adjacency matrix $\mathbf{A}$,
Output: Crisp clustering U.
1: $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{A}$,
2: $[\mathbf{U}_{1,\ldots,K}, \lambda] \leftarrow$ *Lanczos eigensolver* $((\mathbf{L} - \lambda\mathbf{I})\,\mathbf{U} = 0)$
3: $K \leftarrow$ *eigengap*(λ)
4: **return** $\mathsf{U} \leftarrow$ *k-means*$(\mathbf{U}_{1,\ldots,K}, K)$.

2.5.6.2 Normalized Cut

Definition 2.5.30 *(Symmetric (normalized) Graph Laplacian).* Symmetric graph Laplacian matrix [61] is defined as

$$\begin{aligned} \mathbf{L}_{sym} &= \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}, \\ &= \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \end{aligned} \tag{2.180}$$

Definition 2.5.31 *(Random Walk (normalized) Graph Laplacian).* Random walk graph Laplacian matrix [61] resembles the stochastic flows as was previously discussed.

$$\begin{aligned}\mathbf{L}_{rw} &= \mathbf{D}^{-1}\mathbf{L},\\ &= \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}.\end{aligned} \tag{2.181}$$

Theorem 2.9 (Normalized Graph Laplacian and Normalized Cut [90]). *For every vector* $\mathbf{u} \in \mathbb{R}^{|\mathbb{V}|}$*, both* $\mathbf{L}_{sym}$ *and* $\mathbf{L}_{rw}$ *satisfies the following property,*

$$\mathbf{u}^T\mathbf{L}_{sym}\mathbf{u} = \frac{1}{2}\sum_i\sum_j a_{ij}\left(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}}\right)^2, \tag{2.182}$$

which minimization yields the normalized cut of $\mathbb{G}$*.*

Proof: Solving $\mathbf{u}^T\mathbf{L}_{sym}\mathbf{u}$ similarly to Theorem 2.8 we have

$$\begin{aligned}\mathbf{u}^T\mathbf{L}_{sym}\mathbf{u} &= \mathbf{u}^T\mathbf{D}^{-1/2}\mathbf{D}\mathbf{D}^{-1/2}\mathbf{u} - \mathbf{u}^T\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{u},\\ &= \sum_i d_i\left(\frac{u_i}{\sqrt{d_i}}\right)^2 - \sum_i\sum_j a_{ij}\frac{u_i}{\sqrt{d_i}}\frac{u_j}{\sqrt{d_j}},\\ &= \frac{1}{2}\left(\sum_i\sum_j a_{ij}\left(\frac{u_i}{\sqrt{d_i}}\right)^2 - 2\sum_i\sum_j a_{ij}\frac{u_i}{\sqrt{d_i}}\frac{u_j}{\sqrt{d_j}} + \sum_i\sum_j a_{ij}\left(\frac{u_j}{\sqrt{d_j}}\right)^2\right),\\ &= \frac{1}{2}\sum_i\sum_j a_{ij}\left(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}}\right)^2. \qquad \square\end{aligned} \tag{2.183}$$

Recall the objective function of normalized cut,

$$normalizedcut(\mathbb{G}, K) = \text{minimize}\ \sum_i^K \frac{cut(\mathbb{A}_i, \mathbb{G}\backslash\mathbb{A}_i)}{vol(\mathbb{A}_i)},$$

where $\mathbb{A}_i$ denotes the i^{th} subgraph. In normalized cut, the constraint for $\mathbf{U}$ requires the weight of each vertex to be taken into consideration such that $\mathbf{U}^T\mathbf{D}\mathbf{U} = \mathbf{I}$. The minimization is therefore as follows,

$$\begin{aligned}normalizedcut(\mathbb{G}, K) = &\underset{\mathbf{U}}{\text{minimize}} \quad Tr\left(\mathbf{U}^T\mathbf{L}_{sym}\mathbf{U}\right).\\ &\underset{\mathbf{U}}{\text{minimize}} \quad \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{|\mathbb{V}|}\sum_{j=1}^{|\mathbb{V}|} a_{ij}\left(\frac{u_{ik}}{\sqrt{d_i}} - \frac{u_{jk}}{\sqrt{d_j}}\right)^2.\\ &\text{s.t.} \quad \mathbf{U}^T\mathbf{D}\mathbf{U} = \mathbf{I}.\end{aligned} \tag{2.184}$$

Using Lagrange multiplier λ for the constraint, the solution to the minimization problem can be obtained by taking the first partial derivative of the Lagrangian function with respect to $\mathbf{U}$ and setting it to 0 as follows.

$$\begin{aligned} L(\mathbf{U}) &= \mathbf{U}^T\mathbf{L}_{sym}\mathbf{U} - \lambda\left(\mathbf{U}^T\mathbf{D}\mathbf{U} - \mathbf{I}\right), \\ \frac{\partial L}{\partial \mathbf{U}} &= 2\mathbf{L}_{sym}\mathbf{U} - 2\lambda\mathbf{D}\mathbf{U} = 0. \\ \mathbf{L}_{sym}\mathbf{U} &= \lambda\mathbf{D}\mathbf{U}, \end{aligned} \tag{2.185}$$

which is the generalized eigenproblem as in Shi and Malik's proposition [90]. Let $\mathbf{Z} = \mathbf{D}^{1/2}\mathbf{U}$ and $\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ we have the eigensystem of $\mathbf{L}_{rw}$,

$$\begin{aligned} \mathbf{D}^{-1/2}\mathbf{L}\mathbf{Z} &= \lambda\mathbf{D}^{1/2}\mathbf{Z}, \\ \mathbf{D}^{-1}\mathbf{L}\mathbf{Z} &= \lambda\mathbf{Z}, \\ \mathbf{L}_{rw}\mathbf{Z} &= \lambda\mathbf{Z}. \end{aligned} \tag{2.186}$$

Solutions to both Equation 2.185 and Equation 2.186 are obtained by solving the eigenproblems. Similarly to ratio cut, observing the first K eigenvectors starting from the second returns the relaxed solution to the normalized cut. ■

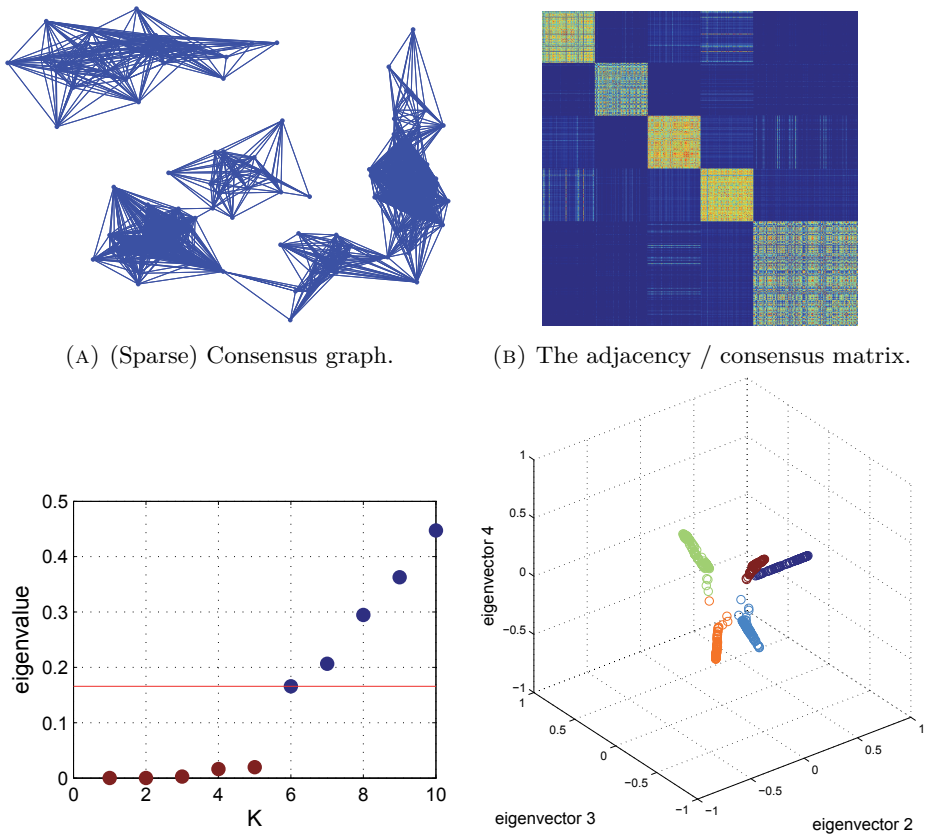
One way to estimate the number of clusters is to observe the *eigengap*, which is the largest deviation between subsequent eigenvalues. Finally, executing k-means on the eigenspace projection easily recover the desired clustering. The normalized cut algorithm pseudocode can be seen in Algorithm 2.12. An illustration of spectral clustering can be seen in Figure 2.18.

Algorithm 2.12 Spectral clustering (normalized cut) algorithm pseudocode

Input: Adjacency matrix $\mathbf{A}$,
Output: Crisp clustering U.
1: $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{A}$,
2: $[\mathbf{U}_{1,\ldots,K}, \lambda] \leftarrow$ *Lanczos eigensolver* $(\mathbf{L}\mathbf{U} = \lambda\mathbf{D}\mathbf{U})$
3: $K \leftarrow$ *eigengap*(λ)
4: **return** $\mathsf{U} \leftarrow$ *k-means*$(\mathbf{U}_{1,\ldots,K}, K)$.

2.6 Consensus Clustering

Following the discussion in the prior sections, we are well aware that divisive algorithms are limited to discovering globular clusters and the capability of graph clustering to recover the information about natural clusters. Even though the formulation of divisive algorithms enables them to assume near to linear complexity, this same formulation also discards every information concerning the



(A) (Sparse) Consensus graph.

(B) The adjacency / consensus matrix.

(C) 5 connected components are found from eigengap analysis.

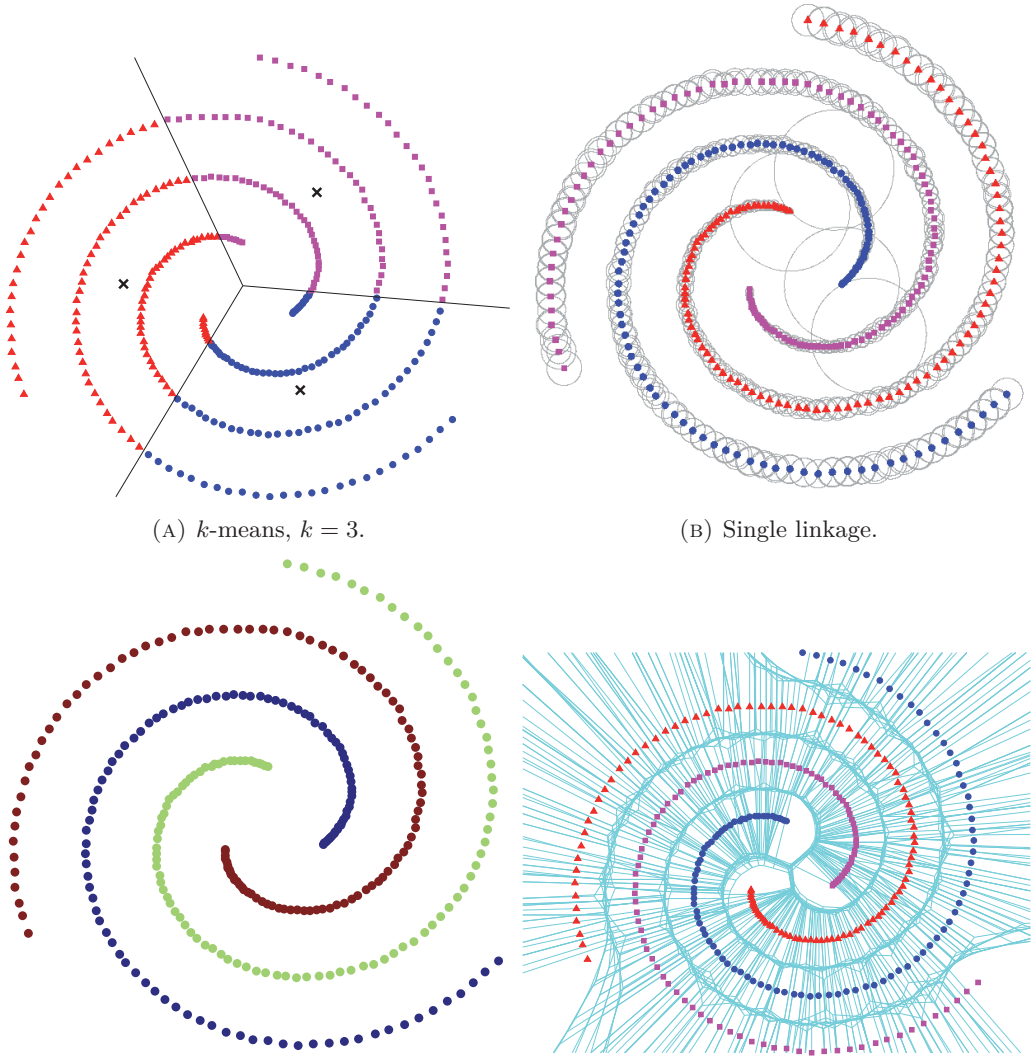
(D) Projection of the graph Laplacian in the eigenspace.

FIGURE 2.18: Illustration of spectral (normalized cut) clustering on the Five Gaussians dataset using the graph obtained from consensus clustering.

inter/intracluster connectivity, which happens to be two important measures for quantifying density [63]. Divisive algorithms also generally fail in discovering the correct cluster in non-convex datasets [3]. Such scenario can be seen in Figure 2.19.

An additional shortcoming common to all the iterative, greedy search-based clustering methods is their sensitivity to the parameter choices and initialization, making the results from divisive clustering inconsistent, therefore harder to trust. Cluster validity measure as discussed in the previous chapter helps to an extent to gain confidence in term of the significance of the divisive clustering result, both in term of number of clusters and correctness of assignments. However, since external objective criterion is generally unavailable in exploratory data analysis, any validity measure, no matter how complete it might seem, would still remain an illusive measure.

Dealing with non-convexity, proximity graph based methods are often preferred over prototype based methods. However, as the amount of data grow, the amount of space and computational power requirement for these methods expands exponentially to a point where execution of the



(A) k-means, $k = 3$. (B) Single linkage.

(C) Spectral Clustering. (D) Ensemble Clustering.

FIGURE 2.19: Various clustering results on non-convex clusters. Prototype based methods experience difficulty on non-convex clusters in general.

algorithm become practically impossible. When clustering data with large volume, analysts are often forced to return to divisive algorithms due to feasibility reasons, even though it lacks the intuitiveness and visual interpretability of Hierarchical clustering [92].

Consensus [92] also called Ensemble [91] Clustering is a relatively new concept in data clustering [91, 92]. The method seeks to benefit from the strengths of both divisive and agglomerative clustering algorithms. Consensus/Ensemble clustering uses numerous monte-carlo runs of divisive clustering algorithms under perturbed data for obtaining the consensus matrix [64, 92]. It has been argued that the natural, and arguably, optimum clusters can be validated with higher confidence by analyzing the stability of this matrix [64, 92]. The consensus matrix provides a more objective

measure of *similarity* from which the natural cluster can be recovered using the appropriate graph theoretic method [64, 91, 107].

2.6.1 Definitions

Definition 2.6.1 *(Consensus Matrix).* Assuming that we have M clustering results from any clustering algorithms, the consensus matrix $\mathcal{C}$ — also referred to as the Co-association matrix in [55, 64, 66, 76, 91] — is an $N \times N$ matrix which describes the accumulated evidence regarding the *stability* of an item in relation to the other items in the data. Figure 2.20 shows the consensus matrices of five Gaussians and uniform data.

Definition 2.6.2 *(Consensus adjacency matrix).* A consensus matrix provides an information regarding item adjacency that can be used in conjunction with an agglomerative hierarchical clustering algorithm to construct a dendrogram. The conversion from consensus matrix $\mathcal{C}$ to consensus adjacency matrix $\mathcal{D}$ is as follows,

$$\mathcal{D} = 1 - \mathcal{C}. \tag{2.187}$$

An illustration of a dendrogram constructed from the consensus adjacency matrix of five Gaussians dataset is shown in Figure 2.20.

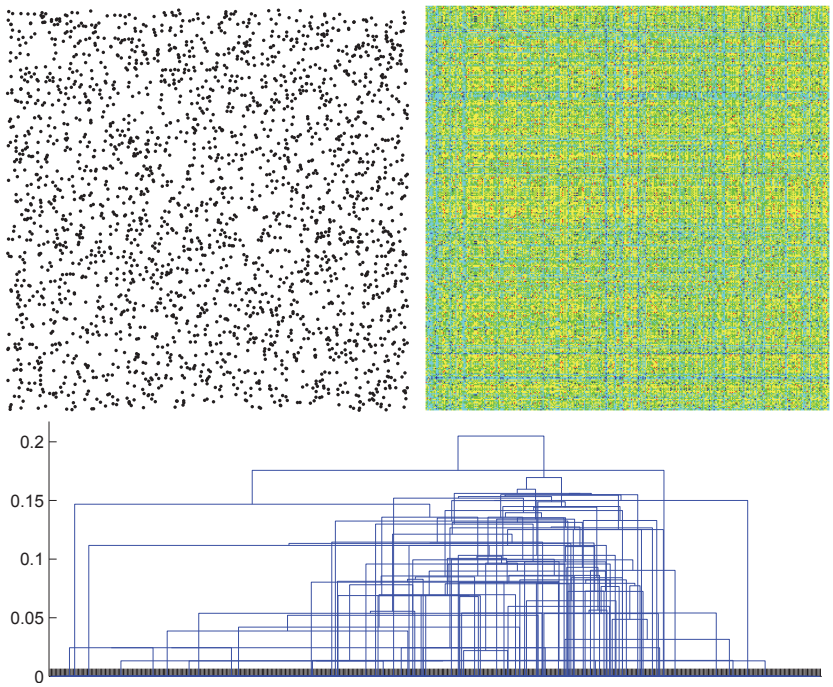
2.6.2 Monti's Consensus Clustering

Monti et al. proposes that robust clusters should be *stable* when subjected to sampling variability. Monti's consensus resampling method constitutes the Consensus matrix from numerous resampling of the data as follows,

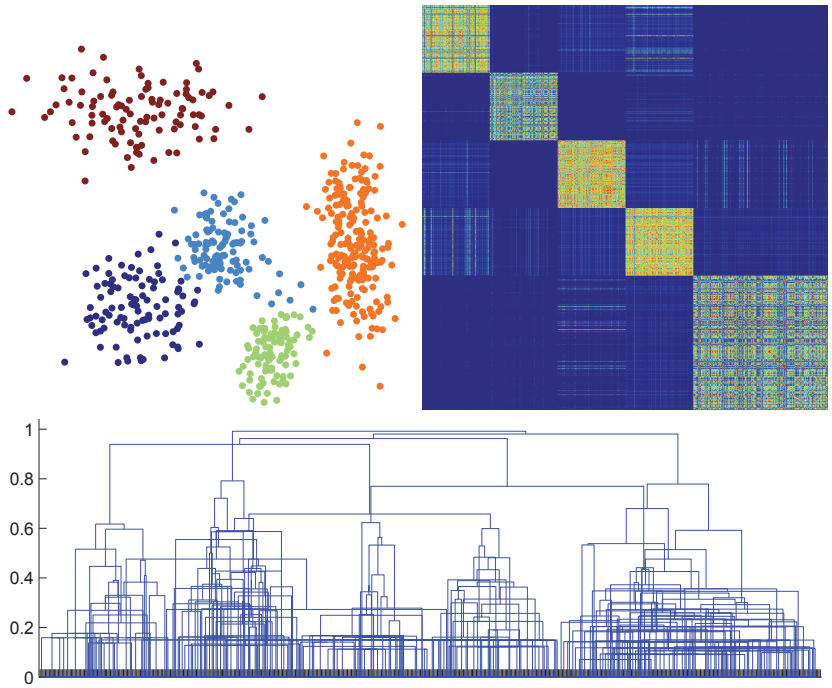
$$\mathcal{C}_{Monti}(s,t) = \frac{\sum_{i=1}^{M} vote(s,t|\mathbb{C}_m)}{\sum_{i=1}^{M} indicator(s,t|\mathbb{X}_m)} \tag{2.188}$$

$$vote(s,t|\mathbb{C}_m) = \begin{cases} 1 & if \quad \text{both item } s \text{ and } t \text{ belong to the same cluster in } \mathbb{C}_m \\ 0 & otherwise \end{cases}, \tag{2.189}$$

$$indicator(s,t|\mathbb{X}_m) = \begin{cases} 1 & if \quad \text{both item } s \text{ and } t \text{ are present in the dataset } \mathbb{X}_m \\ 0 & otherwise \end{cases}, \tag{2.190}$$



(A) Uniform data.



(B) Five Gaussians data.

FIGURE 2.20: The consensus matrices and dendrogram of five Gaussians and uniform data.

where $\mathcal{C}_{Monti(s,t)}$ denotes Monti's consensus index, *vote* denotes the co-occurrence of an item, *indicator* is a boolean matrix which denotes inclusion of a data pair in the random sample $\mathbb{X}_m$, $\mathbb{C}_{m\in\{1,\ldots,M\}}$ denotes clustering results, and $s,t \in \{1,\ldots,N\}$ denotes item indices.

Definition 2.6.3 *(Consensus Stability).* Monti et al. proposes measures that can be used to quantify the stability of each cluster, and to rank items within a given cluster [92]. For each cluster $\mathbb{C}_k$ obtained after k-cuts of the graph constructed from the consensus matrix, the cluster and item stabilities can be calculated as follows,

$$\mathfrak{S}_{cluster}(\mathbb{C}_k) = \frac{1}{|\mathbb{C}_k|(|\mathbb{C}_k|-1)/2} \sum_{\substack{s,t\in\mathbb{C}_k \\ s<t}} \mathcal{C}(s,t), \tag{2.191}$$

$$\mathfrak{S}_{item}(s|\mathbb{C}_k) = \frac{1}{(|\mathbb{C}_k|-1)} \sum_{\substack{t\in\mathbb{C}_k \\ t\neq s}} \mathcal{C}(s,t). \tag{2.192}$$

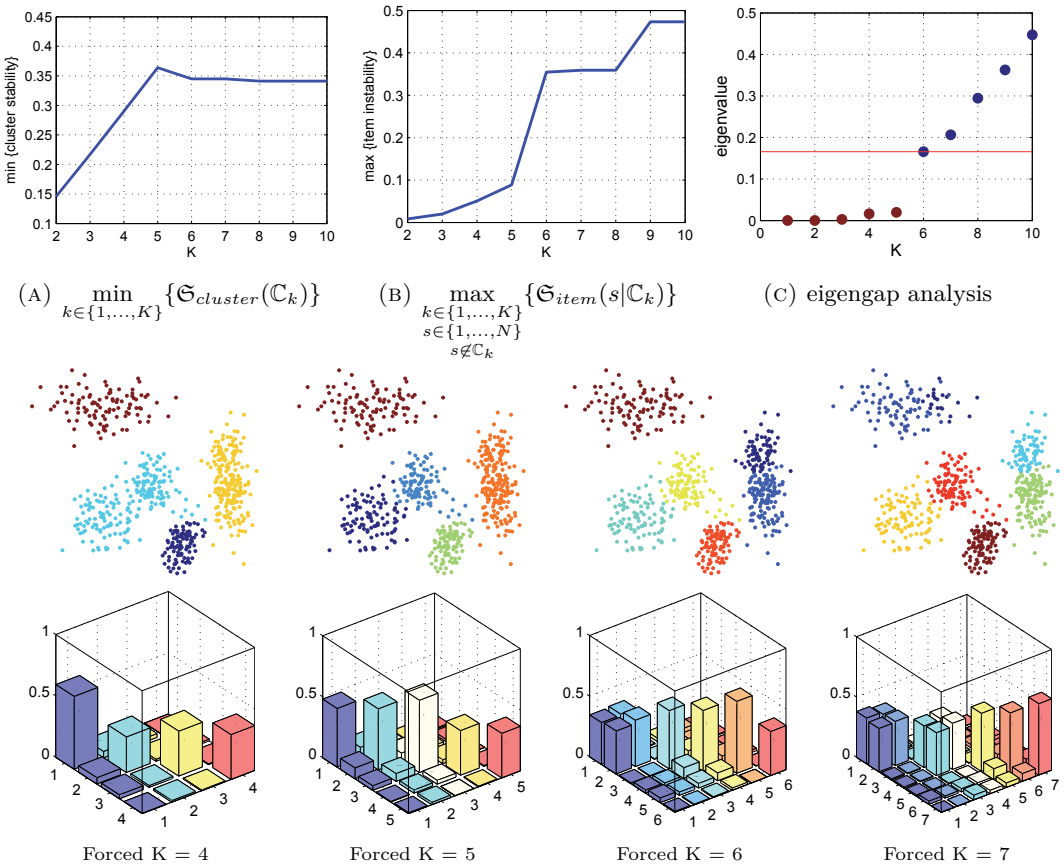
The cluster stability $\mathfrak{S}_{cluster}$ measures the average consensus index between all pairs of items belonging to the same cluster. It operates on the *strictly upper triangular* of the consensus matrix. The item stability $\mathfrak{S}_{item}$ measures the average consensus of an item s relative to all other items in cluster $\mathbb{C}_k$. Stability can be used to determine the appropriateness of a consensus result as shown in Figure 2.21.

Definition 2.6.4 *(Consensus Distribution).* For a given a consensus matrix, the corresponding histogram and *empirical cumulative distribution function CDF* can be calculated as follows,

$$hist(c) = \sum_{s<t} (c-\delta) < \mathcal{C}(s,t) \leq (c+\delta), \tag{2.193}$$

$$CDF(c) = \frac{\sum_{s<t} \mathcal{C}(s,t) \leq c}{N(N-1)/2}. \tag{2.194}$$

Observing the histogram and CDF of a consensus matrix gives an elegant view on the quality of a consensus clustering. A good consensus matrix generally have high concentration of zeros, which indicates good separation between clusters. An illustration is given in Figure 2.22. The area under the curve for uniform data is much lower than Five Gaussians.



(A) $\min_{k\in\{1,\ldots,K\}}\{\mathfrak{S}_{cluster}(\mathbb{C}_k)\}$ (B) $\max_{\substack{k\in\{1,\ldots,K\}\\ s\in\{1,\ldots,N\}\\ s\notin\mathbb{C}_k}}\{\mathfrak{S}_{item}(s|\mathbb{C}_k)\}$ (C) eigengap analysis

(D) Cluster stability matrices for various K. Unstable clusters begin to form at $K > 5$.

FIGURE 2.21: Stability of the five Gaussians data under various K. Item instability sharply increases when K is increased from K=5 to K=6. Consensus stability analysis aligns with the results from eigengap analysis. As instability needs to be minimized, number of clusters above K = 5 would not be desirable.

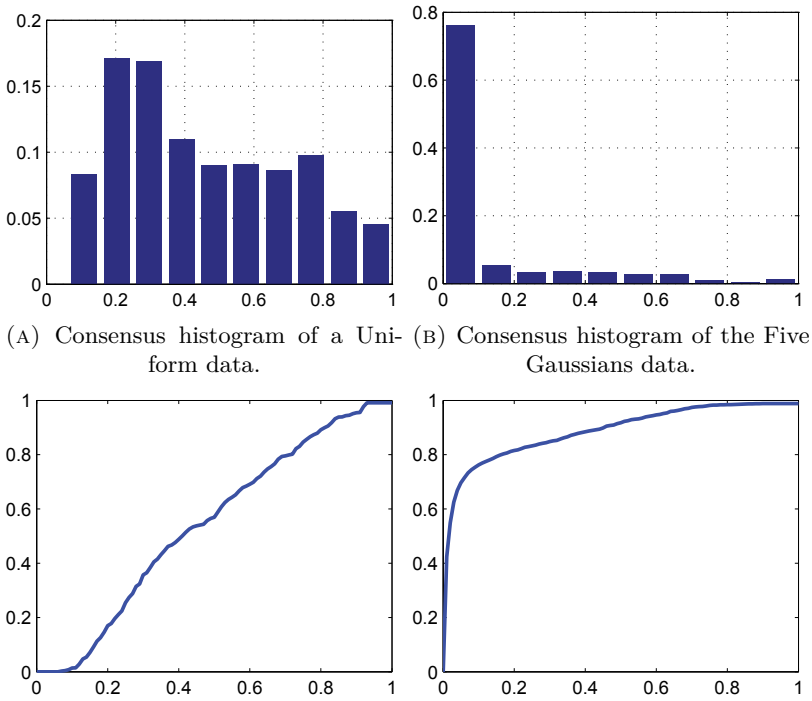
2.6.3 Evidence Accumulation

The Evidence Accumulation (EAC) uses multiple runs of k-means with various K followed by an agglomerative linkage clustering [64, 91]. The consensus matrix is calculated as follows,

$$\mathcal{C}_{EAC}(s,t) = \frac{1}{M}\sum_{m=1}^{M} vote(s,t|\mathbb{C}_m) \tag{2.195}$$

$$vote(s,t|\mathbb{C}_m) = \begin{cases} 1 & if \quad \text{item } s \text{ and } t \text{ belong to the same cluster in } \mathbb{C}_m \\ 0 & otherwise \end{cases}, \tag{2.196}$$

where $\mathbb{C}_{m\in\{1,\ldots,M\}}$ denotes clustering results, and $s,t \in \{1,\ldots,N\}$ denotes item indices.



(A) Consensus histogram of a Uniform data.

(B) Consensus histogram of the Five Gaussians data.

(C) Consensus CDF of a Uniform data. AUC = 0.55.

(D) Consensus CDF of the Five Gaussians data. AUC = 0.89.

FIGURE 2.22: The consensus distribution and CDF of five Gaussians and uniform data.

EAC calculation can be conveniently expressed in term of matrix multiplication as follows,

$$\mathcal{C}_{EAC} = \frac{1}{M} \sum_{m=1}^{M} \mathsf{U_m}^T \mathsf{U_m}. \tag{2.197}$$

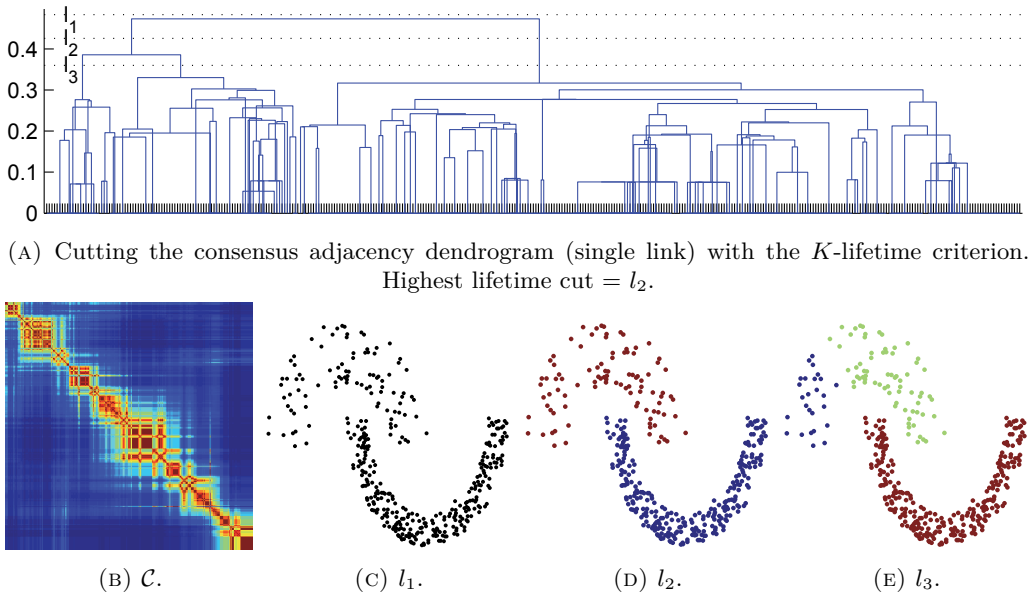
where U_m is a $dim \times N$ matrix which denotes the crisp partitions obtained from the m^{th} run of k-means.

The final partition is then determined using an appropriate linkage clustering (e.g. single/average/-complete/ward linkage). Clusters recovered using single linkage are biased towards connectivity while clusters recovered using average, complete or ward linkage are biased towards tight interconnectedness.

Definition 2.6.5 *(K-lifetime and the highest lifetime criterion).* Fred and Jain define K-cluster lifetime as the range of threshold values on the dendrogram that lead to the identification of K clusters. Applying l_H to the corresponding dendrogram maximizes the average normalized mutual information between partitions [64]. l_H can be formulated as follows,

$$l_H : H = \underset{k,\, k>1}{\arg\max} \{l_k - l_{k-1}\}. \tag{2.198}$$

An illustration can be seen in Figure 2.23.



(A) Cutting the consensus adjacency dendrogram (single link) with the K-lifetime criterion. Highest lifetime cut = l_2.

(B) $\mathcal{C}$. (C) l_1. (D) l_2. (E) l_3.

FIGURE 2.23: Illustration of the K-lifetime criterion on the Half Rings dataset.

2.6.4 Weighted Evidence Accumulation

Weighted Evidence Accumulation (WEAC) was proposed to improve the voting mechanism with the inclusion of internal and relative cluster validity indices to weigh multiple clustering results based on its clustering quality [65]. Given a crisp binary membership matrix from the m^{th} clustering, $\mathsf{U}_m \in [0\,1]$, the co-association matrix is computed as follows,

$$\mathcal{C}_{WEAC} = \frac{\sum_{m=1}^{M} w_m \mathsf{U}_m^T \mathsf{U}_m}{\sum_{m=1}^{M} w_m}, \tag{2.199}$$

where w_m is a scalar denoting the degree of importance (weight) of the m^{th} clustering result.

2.6.5 Fuzzy Evidence Accumulation

Wang proposes the Fuzzy EAC (fEAC) as the extension of EAC for fuzzy clusters [76]. A fuzzy clustering is represented by a fuzzy membership matrix $\mathbf{U}$, where each element u_{tj} defines the membership of the data y_j in the t^{th} cluster. The co-association matrix can be calculated as follows,

$$\mathcal{C}_{fEAC} = \sum_{m=1}^{M} \mathbf{U}_m^T * \mathbf{U}_m, \tag{2.200}$$

where m denotes the clustering solution index. The aggregation product uses the t-norm rule [76],

$$u_{ti,m} * u_{tj,m} = \sum_{t=1}^{k_m} T[u_{ti,m}, u_{tj,m}], \tag{2.201}$$

which is simply the joint probability of the pattern pair i and j over all cluster indices, $t = \{1, \ldots, k_m\}$. k_m denotes the number of clusters in the m^{th} clustering result. Wang suggests that t-norms other than minimum may also be employed [66].

2.6.6 Co-Association Tree

The CA-tree was proposed by Wang in 2011 [66] to create a compact representation of a consensus matrix, extending ensemble algorithms to scale to larger datasets. The CA-tree applies compression to $\mathcal{C}$ in a way that only important representative nodes are retained.

The CA-tree constructs a hierarchical structure similar to a dendrogram using the base cluster label vectors. The CA-tree construction process uses an recursive top-down clustering followed by an efficient bottom-up hamming distance calculation [66]. Pruning this tree at a specific hamming threshold returns the representative nodes which compress the label matrix at a specific rate such that the size of the Consensus matrix can be made smaller. Constructing CA-tree from a label matrix requires very little memory and time complexity and roughly approximates the results of EAC using single linkage [66]. An illustration showing a diagram describing the CA-tree compression on a simple label matrix is shown in Figure 2.24. From this illustration, cutting the tree at th = 0.6 effectively compresses the size of the consensus matrix from 9×9 to 3×3.

2.7 Summary

Clustering is a subjective process of grouping observations into an arbitrary number of meaningful groups subject to further interpretation. In this chapter the general concepts on various paradigm in clustering algorithms and data analysis have been thoroughly revisited. Five integral concepts have been discussed including: Challenges in Clustering algorithms; Various concepts of "distance"; Top-down divisive clustering algorithm; Internal and external cluster validation indices; Graph theoretic algorithms, and; Consensus clustering algorithms.

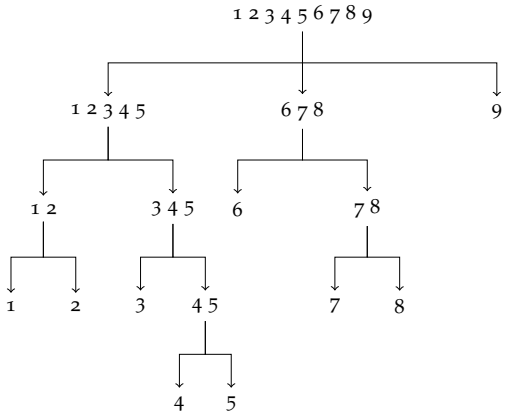
Each clustering paradigm comes with its own advantage and disadvantage. Relative to graph algorithms, divisive algorithms are generally more lightweight ($\mathcal{O}(N)$) which make them easily applicable to larger datasets. Drawbacks of divisive algorithms are: 1. They are limited to form

	1	2	3	4	5	6	7	8	9
$\mathbb{C}_1$	1	1	1	1	1	2	2	3	2
$\mathbb{C}_2$	1	1	2	2	2	3	2	2	3
$\mathbb{C}_3$	1	2	2	1	2	2	2	3	3
$\mathbb{C}_4$	4	4	4	4	2	2	3	1	3

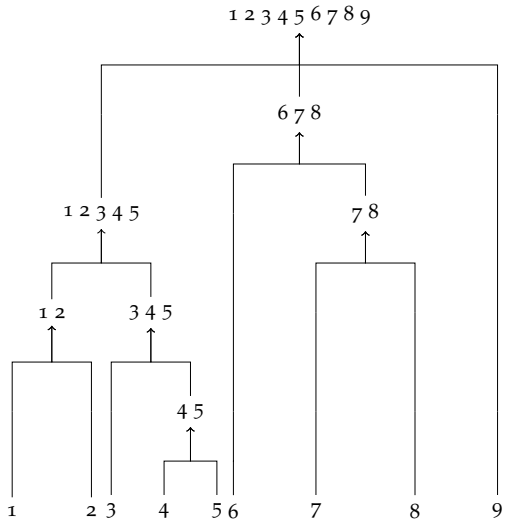
(A) Label matrix & desired compression.

	r_1	r_2	r_3
$\mathbb{C}_1$	1	3	2
$\mathbb{C}_2$	2	2	3
$\mathbb{C}_3$	2	3	3
$\mathbb{C}_4$	4	1	3

(B) Representative nodes. th = 0.6.



(C) Depth tree obtained from top-down division on the label matrix.



(D) Augmented depth tree after recursive bottom-up distance calculation.

FIGURE 2.24: Illustration of the CA-tree.

clusters around prototypes; 2. The appropriate number of prototypes is subjective and has to be inferred using the appropriate cluster validation index; 3. Solutions are subject to suboptimality as they are dependent on the initial positions of the prototypes.

The first and second drawbacks in the divisive clustering are answered in graph algorithms. Connected graph offers useful heuristics for determining the appropriate number of clusters at a significantly more expensive price ($\mathcal{O}(N^2)$ compared to $\mathcal{O}(N)$). This complexity can make graph algorithms difficult to afford, especially when they are executed using a machine with limited processing power or memory.

Consensus clustering algorithms combine the paradigms of both divisive and graph theoretic. It is argued that a relatively more robust clustering can be achieved using consensus clustering paradigms. Having graph algorithm as part of the algorithmic constituent, the complexity of consensus algorithm is lower bounded to $\mathcal{O}(N^2)$ which makes them rather unsuitable to be used for large data.

Chapter 3

Particle Swarm Optimization and Data Clustering

Particle Swarm Optimization (PSO) is a parallel evolutionary computation technique for general nonlinear function optimization first proposed by Kennedy and Eberhart in 1995 [154], which is based on a social behavior metaphor. The PSO algorithm is initialized randomly with candidate solutions, conceptualized as particles. Each particle is assigned a randomized velocity and is iteratively repositioned through the problem space. It is attracted by the location of its *personal best* fitness and the swarm *local/global best* fitness. Since its proposition, the standard PSO algorithm has been through continuous improvement and analysis. It has also inspired a plethora of other PSO variants customized for various purposes and applications, including cluster optimization [4, 55, 70, 74, 75, 83].

This chapter is organized as follows: Section 3.1 introduces the basic terminologies, analysis of convergence of PSO where a discourse against Jiang et al.'s stability conditions is provided, and complexity of the PSO algorithm; Section 3.2 introduces Van Der Merwe and Engelbrecht's PSO clustering framework; Section 3.3 argues against the optimality of the PSO clustering algorithm, where both theoretical and empirical proofs have been provided to support the argument; Finally Section 3.4 provides the concluding remarks for the chapter.

3.1 Particle Swarm Optimization

Definition 3.1.1 *(Objective function).* An objective function is a mathematical equation that is to be optimized (either minimized or maximized) given a certain constraints.

Definition 3.1.2 *(Search Space).* A search space Ω is a predefined constraints describing the feasible boundary for the input of the objective function.

Definition 3.1.3 *(Particle).* A particle θ stores a position vector $\mathbf{x}$, a velocity vector $\mathbf{v}$, and a personal best vector $\mathbf{p}$,

$$\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\}. \tag{3.1}$$

Definition 3.1.4 *(Swarm).* A swarm is a set of particles,

$$\mathbf{\Theta} = \{\theta_1, \ldots, \theta_K\}. \tag{3.2}$$

Definition 3.1.5 *(Position).* The position of a particle $\mathbf{x}$ encodes a set of parameters which represents a possible solution to the optimization problem which is updated as follows,

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1), \tag{3.3}$$

where $\mathbf{v}$ is the velocity vector which is updated according to the personal best and local/global best positions.

Definition 3.1.6 *(Personal Best).* Each particle has a memory of its personal best position $\mathbf{p}$, which goodness is determined by how well it optimizes the objective function $f(\mathbf{x})$. In the case where $f(\mathbf{x})$ needs to be minimized, the personal best is updated as follows,

$$\mathbf{p}(t+1) = \begin{cases} \mathbf{x}(t) & if\ f(\mathbf{x}(t)) < f(\mathbf{p}(t)) \\ \mathbf{p}(t) & otherwise \end{cases} \tag{3.4}$$

Definition 3.1.7 *(Local Best).* The local best $\mathbf{l}$ is the best position vector of the local particles, which is shared locally among neighboring particles in the local neighborhood $\mathbb{L}$. In the case where $f(\mathbf{x})$ needs to be minimized, and given a best position of the neighboring particle $\mathbf{p}_n \in \mathbb{L}$, the local best $\mathbf{l}$ of $\mathbb{L}$ is updated as follows,

$$\mathbf{l}(t+1) = \begin{cases} \mathbf{p}_n(t) & if\ f(\mathbf{p}_n(t)) < f(\mathbf{l}(t)) \\ \mathbf{l}(t) & otherwise \end{cases}. \tag{3.5}$$

A popular neighborhood topology for PSO is the ring topology. The local best PSO is particularly useful for multiobjective optimization where one needs to discover as many local optima as possible (e.g. niching strategy) [155].

Definition 3.1.8 *(Global (swarm) Best).* The global best $\mathbf{g}$ is a best position vector of the swarm, which is shared globally among all particles in the swarm. In the case where $f(\mathbf{x})$ needs to be minimized, the global best is updated as follows,

$$\mathbf{g}(t+1) = \begin{cases} \mathbf{p}(t) & if\ f(\mathbf{p}(t)) < f(\mathbf{g}(t)) \\ \mathbf{g}(t) & otherwise \end{cases}, \tag{3.6}$$

Definition 3.1.9 *(Velocity).* The velocity of a particle $\mathbf{v}$ describes its movement trajectory. For each particle $\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\}$ in a swarm $\Theta = \{\theta_1, \ldots, \theta_M; \mathbf{g}\}$ (global best), and a local best $\mathbf{l}$, the update equation for PSO can be described as follows,

Global best PSO

$$\mathbf{v}(t+1) = \omega(t)\mathbf{v}(t) + c_p\varphi_p \circ (\mathbf{p}(t) - \mathbf{x}(t)) + c_g\varphi_g \circ (\mathbf{g}(t) - \mathbf{x}(t)), \tag{3.7}$$

Local best PSO

$$\mathbf{v}(t+1) = \omega(t)\mathbf{v}(t) + c_p\varphi_p \circ (\mathbf{p}(t) - \mathbf{x}(t)) + c_l\varphi_l \circ (\mathbf{l}(t) - \mathbf{x}(t)), \tag{3.8}$$

where $\omega(t)$ is the momentum function, c_p denotes the cognitive constant, c_g denotes the social constant, c_l denotes the social (local) constant, $\varphi_{p,l,g} \in \{0, 1\} \in \mathbb{R}^{dim}$ denote uniform random numbers, and $\circ$ denotes Hadamard product. High ω (e.g. $\{0.5 - 1\}$) encourages exploration, while low ω (e.g. close to 0) encourages exploitation.

To avoid swarm explosion, the velocity is constricted at a predefined ratio $\eta\%$ of the search space Ω as follows,

$$\mathbf{v}(t) = \max(\min(\mathbf{v}(t), \mathbf{v}_{\max}), -\mathbf{v}_{\max}), \tag{3.9}$$

$$\mathbf{v}_{max} = \eta\% \cdot \Omega. \tag{3.10}$$

Since both Equation 3.7 and Equation 3.8 are biased towards axis-parallel problems [156, 157], Zambrano-Bigiarini et al. proposed the Standard PSO (SPSO) 2011 which takes into account the geometrical attributes of cross-dimensional movements to make the swarm invariant to axis

rotations. The velocity update scheme for SPSO 2011 is as follows,

$$\mathbf{x}_{\overrightarrow{xp}}(t) = \mathbf{x}(t) + c_p\varphi_p \circ (\mathbf{p}(t) - \mathbf{x}(t)) \tag{3.11}$$

$$\mathbf{x}_{\overrightarrow{xl}}(t) = \mathbf{x}(t) + c_p\varphi_p \circ (\mathbf{l}(t) - \mathbf{x}(t)) \tag{3.12}$$

$$\mathfrak{G}(t) = \frac{\mathbf{x}(t) + \mathbf{x}_{\overrightarrow{xp}}(t) + \mathbf{x}_{\overrightarrow{xl}}(t)}{3} \tag{3.13}$$

$$\mathbf{v}(t+1) = \omega\mathbf{v}(t) + \mathcal{H}(\mathfrak{G}(t), ||\mathfrak{G}(t) - \mathbf{x}(t)||) - \mathbf{x}(t), \tag{3.14}$$

where $\mathcal{H}(\mathfrak{G}(t), ||\mathfrak{G}(t) - \mathbf{x}(t)||)$ denotes a random vector drawn inside the hypersphere with radius $||\mathfrak{G}(t) - \mathbf{x}(t)||$, centered at $\mathfrak{G}$.

The generic PSO pseudocode is shown in Algorithm 3.13

Algorithm 3.13 Particle Swarm Optimization: generic pseudocode

Input: Objective function $f(\mathbf{x})$, search space $\Omega = [\mathbf{x}_{min}, \mathbf{x}_{max}]$, number of swarms M, constants c_p, c_l, c_g, momentum function $\omega(t)$, velocity constriction $[\mathbf{v}_{min}, \mathbf{v}_{max}]$, neighborhood $\mathbb{L}$, maximum number of iteration t_{max}.
Output: The swarm Θ.

for all $\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\} \in \Theta$
  $\mathbf{x} \leftarrow rand(\Omega)$
  $\mathbf{v} \leftarrow \{0\}$
end for
while $t < t_{max}$
  for all $\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\} \in \Theta \in \mathbb{L}$
    $\mathbf{v} \leftarrow update(\mathbf{v})$
    $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$
    $\mathbf{p} = \arg\min_{\mathbf{x}} f(\mathbf{x})$
    $\mathbf{l} = \arg\min_{\mathbf{p}_n} f(\mathbf{p}_n)$ (for global best, use star neighborhood)
  end for
end while
return Θ

3.1.1 Convergence Analysis

In depth convergence analysis of PSO has been given by a number of authors including Jiang et al. [81] and Clerc and Kennedy [82]. This section analyzes stability and convergence from control theory point of view.

For simplicity, let us observe the trajectory of the PSO on specifically one of the dimensions. The overall trajectory of a specific particle can be summarized as follows.

$$\begin{aligned} v(t+1) &= \omega v(t) + c_p\varphi_p(p(t) - x(t)) + c_g\varphi_g(g(t) - x(t)), \\ &= \omega v(t) - (c_p\varphi_p + c_g\varphi_g)x(t) + c_p\varphi_p p(t) + c_g\varphi_g g(t), \\ x(t+1) &= x(t) + v(t+1). \end{aligned} \tag{3.15}$$

Let $\mathbf{X}(t) = [x(t), v(t)]^T$ and $\mathbf{U}(t) = [p(t), g(t)]^T$, Equation 3.15 can expressed in an explicit discrete time-variant state space format,

$$\begin{bmatrix} x(t+1) \\ v(t+1) \end{bmatrix} = \overbrace{\begin{bmatrix} 1 - (c_p\varphi_p + c_g\varphi_g) & \omega \\ -(c_p\varphi_p + c_g\varphi_g) & \omega \end{bmatrix}}^{\mathbf{A}(t)} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \overbrace{\begin{bmatrix} c_p\varphi_p & c_g\varphi_g \\ c_p\varphi_p & c_g\varphi_g \end{bmatrix}}^{\mathbf{B}(t)} \begin{bmatrix} p(t) \\ g(t) \end{bmatrix} \tag{3.16}$$

$$\mathbf{Y}(t) = \overbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}^{\mathbf{C}} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}, \tag{3.17}$$

Obviously, the PSO system is time variant due to the influence of the uniform random variables $\varphi_p \in \{0, 1\}$ and $\varphi_g \in \{0, 1\}$. $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are equally likely to assume any of the matrix in the infinite set,

$$\mathbb{A} = \left\{ \overleftarrow{(\varphi_p, \varphi_g) \to \inf(\varphi) = 0} \quad \overrightarrow{(\varphi_p, \varphi_g) \to \sup(\varphi) = 1} \right. \\ \left. \overbrace{\begin{bmatrix} 1 & \omega \\ 0 & \omega \end{bmatrix}}^{\inf(\mathbf{A})}, \ldots, \begin{bmatrix} 1 - (c_p\varphi_p + c_g\varphi_g) & \omega \\ -(c_p\varphi_p + c_g\varphi_g) & \omega \end{bmatrix}, \ldots, \overbrace{\begin{bmatrix} 1 - (c_p + c_g) & \omega \\ -(c_p + c_g) & \omega \end{bmatrix}}^{\sup(\mathbf{A})} \right\} \tag{3.18}$$

$$\mathbb{B} = \left\{ \overleftarrow{(\varphi_p, \varphi_g) \to \inf(\varphi) = 0} \quad \overrightarrow{(\varphi_p, \varphi_g) \to \sup(\varphi) = 1} \right. \\ \left. \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}}^{\inf(\mathbf{B})}, \ldots, \begin{bmatrix} c_p\varphi_p & c_g\varphi_g \\ c_p\varphi_p & c_g\varphi_g \end{bmatrix}, \ldots, \overbrace{\begin{bmatrix} c_p & c_g \\ c_p & c_g \end{bmatrix}}^{\sup(\mathbf{B})} \right\} \tag{3.19}$$

Due to this randomness, giving a complete analysis on the PSO is difficult and still subject to much research. For example Clerc and Kennedy elaborates in great detail regarding the occurrence of swarm explosion due to this random weighting [82].

Lemma 3.1.1 *(Parameter Selection, Swarm Stability and Convergence of PSO).* ***PSO is guaranteed to converge to an equilibrium point when the stability criteria,***

$$0 \leq \omega < 1, \text{ and } 0 < c_p + c_g < 2 + 2\omega, \tag{3.20}$$

are satisfied.

Proof: By deriving the stability criterion, we can choose the parameters that ensure convergence. The assumption in this analysis is that $0 < (\varphi_p, \varphi_g) \leq 1$ such that $\mathbf{U}(t)$ is always reachable in a finite number of iterations. On this regard, we discover some limitations in the work of Jiang et al. [81]. Jiang et al. mistakenly generalizes the system response using expected values, simplifying the random numbers to $E[\varphi_p] = E[\varphi_g] = 0.5$ [81]. Under this assumption, the stability of the system is guaranteed only when $0 < (\varphi_p, \varphi_g) \leq 0.5$ but not when $0.5 < (\varphi_p, \varphi_g) \leq 1$.

Therefore, we propose representing φ using the supremum of the set,

$$\sup(\varphi_p) = \sup(\varphi_g) = 1, \tag{3.21}$$

at which the control energy is at the highest. This way the stability can be ensured for the range $0 < \varphi \leq 1$.

Representing the random numbers with their suprema we have,

$$\begin{bmatrix} x(t+1) \\ v(t+1) \end{bmatrix} = \overbrace{\begin{bmatrix} 1-(c_p+c_g) & \omega \\ -(c_p+c_g) & \omega \end{bmatrix}}^{\sup(\mathbf{A})} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \overbrace{\begin{bmatrix} c_p & c_g \\ c_p & c_g \end{bmatrix}}^{\sup(\mathbf{B})} \begin{bmatrix} p(t) \\ g(t) \end{bmatrix} \tag{3.22}$$

The transfer function $\mathbf{H}(z)$ using the supremum is as follows [158],

$$\begin{aligned} \mathbf{H}(z) = \frac{\mathbf{Y}(z)}{\mathbf{U}(z)} &= \mathbf{C}(z\mathbf{I} - \sup(\mathbf{A}))^{-1}\sup(\mathbf{B}) \\ &= \frac{\mathbf{C}\, adj(z\mathbf{I} - \sup(\mathbf{A}))\ \sup(\mathbf{B})}{|z\mathbf{I} - \sup(\mathbf{A})|}, \end{aligned} \tag{3.23}$$

which satisfies stability criterion when the poles are inside the unit circle, $0 < |z_{1,2}| < 1$. In other words, the eigenvalues of $\sup\{\mathbf{A}\}$ should be $0 < |\lambda_{1,2}| < 1$. Solving $|z\mathbf{I} - \sup(\mathbf{A})|$ we have the characteristic equation,

$$|z\mathbf{I} - \sup(\mathbf{A})| = z^2 + (c_p + c_g - \omega - 1)\, z + \omega. \tag{3.24}$$

The PSO transfer function is then,

$$\mathbf{H}(z) = \frac{1}{z^2 + (c_p + c_g - \omega - 1)\, z + \omega} \begin{bmatrix} c_p z & c_g z \end{bmatrix} \tag{3.25}$$

Using the quadratic equation property and imposing the stability constraint, the poles of the PSO transfer function can be calculated,

$$|z_{1,2}| = \left| \frac{1 + \omega - (c_p + c_g) \pm \sqrt{(c_p + c_g - \omega - 1)^2 - 4\omega}}{2} \right| < 1. \tag{3.26}$$

Consider two cases:

1. **Poles contain complex numbers:** When $(c_p + c_g - \omega - 1)^2 < 4\omega$ we have an upper bound,

$$c_p + c_g < 1 + \omega \pm 2\sqrt{\omega}. \tag{3.27}$$

   Since both c_p and c_g are non-negative, nonzero real numbers, it must follow that $\omega > 0$. Constraining the diameter of Equation 3.26 to the unit circle we have another upper bound,

$$c_p + c_g < 1 + \omega \pm \sqrt{2 + 2\omega}. \tag{3.28}$$

   Synthesizing both constraints gives $c_p + c_g < 1 + \omega \pm 2\sqrt{\omega} < 1 + \omega \pm \sqrt{2 + 2\omega}$, which is feasible when $0 < \omega < 1$. The least upper bound for $c_p + c_g$ from case 1 is therefore $1 + \omega \pm 2\sqrt{\omega}$. Thus, from the first case we get

$$0 < \omega < 1 \text{ and } 1 + \omega - 2\sqrt{\omega} < c_p + c_g < 1 + \omega + 2\sqrt{\omega}. \tag{3.29}$$

2. **Poles contain only real numbers:** When $(c_p + c_g - \omega - 1)^2 \geq 4\omega$ we have,

$$c_p + c_g \leq 1 + \omega - 2\sqrt{\omega} \text{ or } c_p + c_g \quad \geq 1 + \omega + 2\sqrt{\omega} \tag{3.30}$$

   Similarly as case 1., since both c_p and c_g are real numbers, it must follow that $\omega \geq 0$. Constraining the diameter by 1 gives us two constraints on the poles,

$$z_1 = \frac{1}{2}\left(1 + \omega - (c_p + c_g) + \sqrt{(c_p + c_g - \omega - 1)^2 - 4\omega}\right) < 1, \text{ and} \tag{3.31}$$

$$z_2 = \frac{1}{2}\left(1 + \omega - (c_p + c_g) - \sqrt{(c_p + c_g - \omega - 1)^2 - 4\omega}\right) > -1. \tag{3.32}$$

Solving z_1 and imposing $\omega \geq 0$ gives

$$0 < c_p + c_g \leq 1 + \omega - 2\sqrt{\omega}, \text{which is always true when } \omega \neq 1. \tag{3.33}$$

Imposing the feasible region for ω for case 1., we then have $0 \leq \omega < 1$.

Continuing, solving z_2 gives,

$$c_p + c_g < 2 + 2\omega, \tag{3.34}$$

Synthesizing the constraints with Equation 3.30, the feasible boundary condition for ω and $c_p + c_g$ we obtain,

$$0 < c_p + c_g < 2 + 2\omega. \tag{3.35}$$

Synthesizing both cases we have the convergence boundary for the whole range of uniform random numbers $0 \leq \varphi \leq 1$ which is,

$$0 \leq \omega < 1, \text{ and } 0 < c_p + c_g < 2 + 2\omega. \tag{3.36}$$

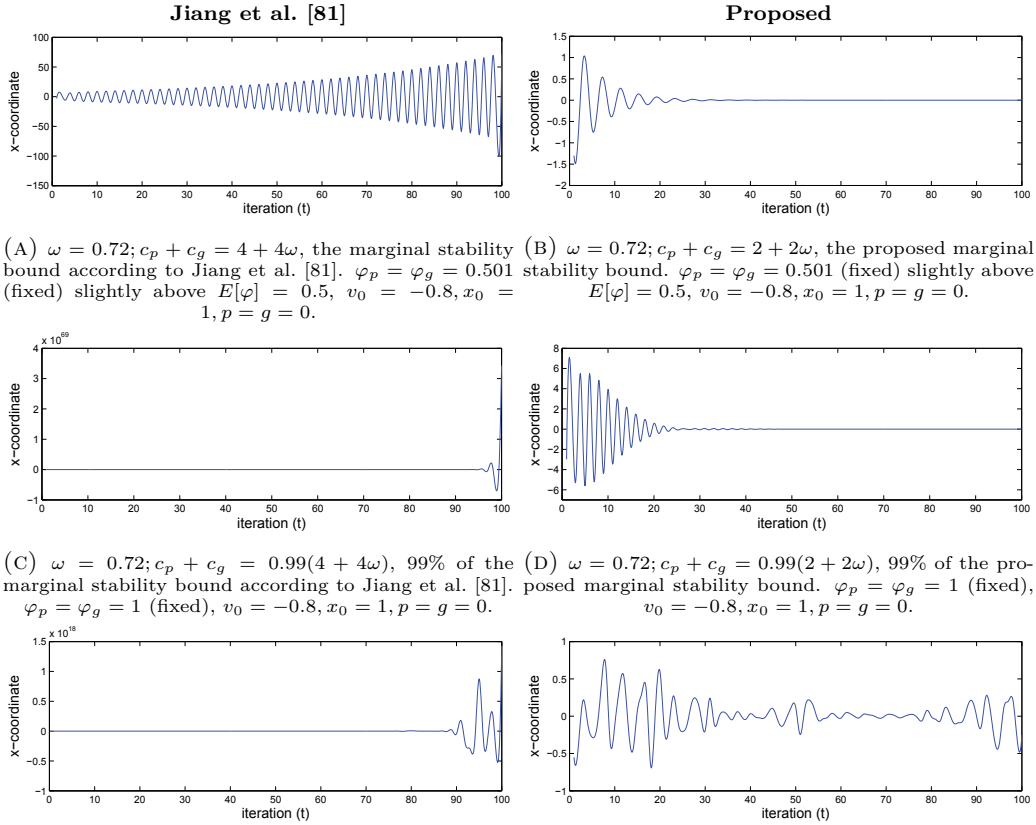
We observe that Equation 3.36 imposes a least upper bound for $c_p + c_g$ that is twice lower than Jiang et al.'s boundary ($0 \leq \omega < 1, \text{ and } 0 < c_p + c_g < 4 + 4\omega.$) [81]. This stricter bounds imposes important characteristic that ensures stability when the values of both random weightings are higher than their expected values.

Without loss of generality, the inputs to the system (personal best and global best positions) can be represented as two step functions with gains of P and G. Applying final value theorem [158] to the PSO transfer function completes the analysis.

$$\begin{aligned} \lim_{t\to\infty} x(t) &= \lim_{z\to 1} \left\{(z-1)\mathbf{H}(z)\mathbf{U}(z)\right\}, \\ &= \lim_{z\to 1} \left\{ \frac{(c_p P + c_g G)z^2}{z^2 + (c_p + c_g - \omega - 1)\, z + \omega} \right\}, \\ &= \frac{c_p P + c_g G}{c_p + c_g}, \end{aligned} \tag{3.37}$$

which is consistent with the earlier reports by both [82] and [81].

Empirical results comparing Jiang et al.'s [81] and our proposed stability bounds are shown in Figure 3.1. We can observe that Jiang et al.'s stability criterion causes the particle to diverge at $\varphi > 0.5$. The particle do not converge even under randomized φ using Jiang et al.'s marginal stability criterion. Our proposed bounds generally permit the particle to converge.



(A) $\omega = 0.72; c_p + c_g = 4 + 4\omega$, the marginal stability bound according to Jiang et al. [81]. $\varphi_p = \varphi_g = 0.501$ (fixed) slightly above $E[\varphi] = 0.5$, $v_0 = -0.8, x_0 = 1, p = g = 0$.

(B) $\omega = 0.72; c_p + c_g = 2 + 2\omega$, the proposed marginal stability bound. $\varphi_p = \varphi_g = 0.501$ (fixed) slightly above $E[\varphi] = 0.5$, $v_0 = -0.8, x_0 = 1, p = g = 0$.

(C) $\omega = 0.72; c_p + c_g = 0.99(4 + 4\omega)$, 99% of the marginal stability bound according to Jiang et al. [81]. $\varphi_p = \varphi_g = 1$ (fixed), $v_0 = -0.8, x_0 = 1, p = g = 0$.

(D) $\omega = 0.72; c_p + c_g = 0.99(2 + 2\omega)$, 99% of the proposed marginal stability bound. $\varphi_p = \varphi_g = 1$ (fixed), $v_0 = -0.8, x_0 = 1, p = g = 0$.

(E) $\omega = 0.72; c_p + c_g = 4 + 4\omega$, the marginal stability bound according to Jiang et al. [81]. $\varphi_p = \varphi_g = \{0, 1\}$ (random), $v_0 = -0.8, x_0 = 1, p = g = 0$.

(F) $\omega = 0.72; c_p + c_g = 2 + 2\omega$, the proposed marginal stability bound. $\varphi_p = \varphi_g = \{0, 1\}$ (random), $v_0 = -0.8, x_0 = 1, p = g = 0$.

FIGURE 3.1: Convergence of a particle on 1-dimensional optimization problem with various stability criteria. Plots are interpolated using spline interpolant for smoother visualization.

3.1.2 Complexity and Performance Analysis

PSO is generally a lightweight and simplistic algorithm, and hence makes it very attractive for general stochastic nonlinear optimization. The update complexity of PSO is linear to the number of dimensions and particles [77]. From the state space formula in Equation 3.16, we can easily show that for each dimension, the update equation of PSO can be expressed a simple matrix multiplication and addition. The main contributor to PSO complexity is therefore the cost for evaluating the given objective function, which varies depending on the application.

Its performance were also shown to be comparable to that of the genetic algorithm (GA) [159]. PSO is claimed to converge to local optimum faster than GA on a number of low dimensional problems [159].

3.2 Van Der Merwe - Engelbrecht's PSO Clustering

The earliest proposition of PSO-based clustering algorithm was by Van Der Merwe and Engelbrecht in 2003 [83].

Definition 3.2.1 *(data vector).* A data vector or observation $\mathbf{y} \in \mathbb{R}^{dim}$ is represented as a *dim*-dimensional column vector.

Definition 3.2.2 *(position).* The position vector of each particle in PSO clustering represents a set of all K potential centroid vectors e.g. $\mathbf{x}_i = \{\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,K}\} \in \mathbb{R}^{dim}$.

The swarm's initial position can be either seeded with the centroid vectors obtained with k-means or initialized at random. The goal of the swarm is then to improve the initial position vectors and find a global best $\mathbf{g}$ that minimizes the average quantization error,

$$\mathbf{g} = \arg\min_{\mathbf{x}} f(\mathbf{x}), \tag{3.38}$$

$$f(\mathbf{x}) = \frac{1}{K} \sum_{\mathbf{z} \in \mathbf{x}} \sum_{\mathbf{y} \in \mathbb{C}_z} \frac{1}{|\mathbb{C}_z|} d(\mathbf{y}, \mathbf{z}), \tag{3.39}$$

where the cluster assignment for each data follows the framework of k-means wherein data are crisply partitioned, obeying voronoi tessellation principles. The pseudocode is shown in Algorithm 3.14.

Algorithm 3.14 Van Der Merwe - Engelbrecht's PSO Clustering

Input: A set of data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, Number of clusters K, search space $\Omega = [\mathbf{x}_{min} = \lfloor \mathbb{Y} \rfloor, \mathbf{x}_{max} = \lceil \mathbb{Y} \rceil]$, constants c_p, c_g, velocity clamp $[\mathbf{v}_{min}, \mathbf{v}_{max}]$, maximum iteration t_{max}.
Output: Global best centroid vector $\mathbf{g}$.

for all $\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\} \in \Theta$
  $\mathbf{x} \leftarrow k\text{-}means(\mathbb{Y}, K)$ or $\mathbf{x} \leftarrow rand(\Omega)$
  $\mathbf{v} \leftarrow \{0\}$
end for
while $t < t_{max}$
  for all $\theta = \{\mathbf{x}, \mathbf{v}, \mathbf{p}\} \in \Theta$
    $\mathbf{v} \leftarrow update(\mathbf{v})$
    $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$
    $\mathbb{Z} \leftarrow reshape(\mathbf{x}, [K, dim])$
    $D \leftarrow D(\mathbb{Y}, \mathbb{Z})$
    $\forall j = \{j : 1, \ldots, N\},\ \mathbb{C}_I \leftarrow \mathbf{y}_j,\ where\ I = \arg\min_i D_{ij}$
    $\mathbf{p} = \arg\min_{\mathbf{x}} f(\mathbf{x})$
    $\mathbf{g} = \arg\min_{\mathbf{p}} f(\mathbf{p})$
  end for
end while
return g

The prototype trajectory: initialization and result for a simple 2 dimensional clustering case is illustrated in Figure 3.2.

3.2.1 Performance Analysis

As the update complexity of the swarm is lightweight by nature, complexity of Van Der Merwe - Engelbrecht's PSO Clustering mainly arises from the distance matrix calculation $\mathcal{O}(KN)$ for each particle on each iteration. Hence the overall complexity of the algorithm on each iteration is $\mathcal{O}(MKN)$, where M denotes the number of particles in the swarm. The complexity of an execution over T iterations is $\mathcal{O}(MKNT)$, which is linear to the number of data.

Despite its linear complexity, the algorithm requires a significantly larger amount of function evaluation to achieve convergence compared to its deterministic counterpart (k-means) due to Section 3.3.

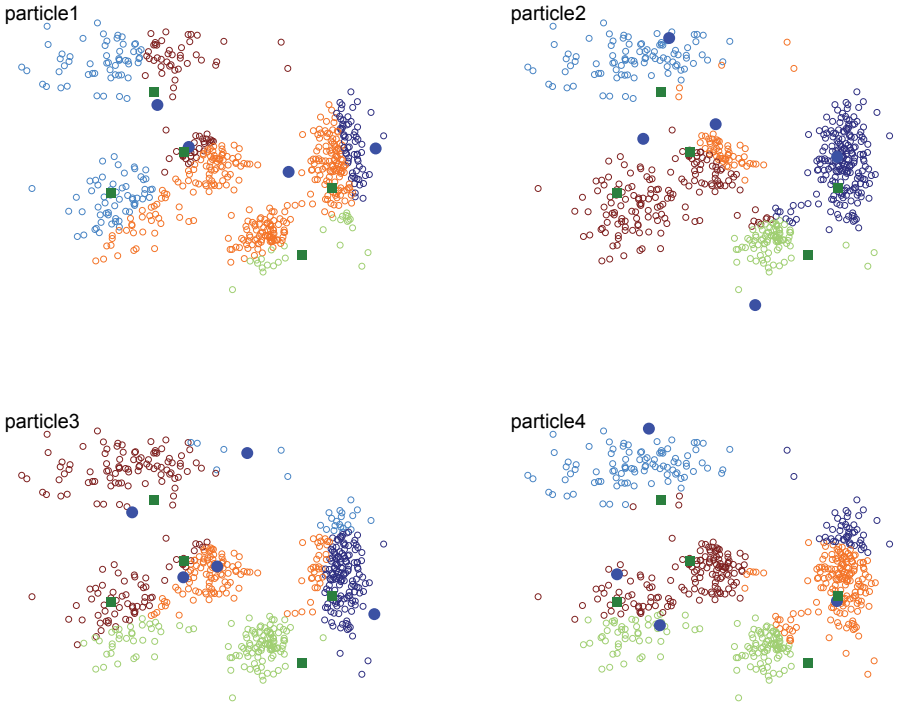
Van Der Merwe and Engelbrecht claims that the PSO clustering generally produces better outcome than Stuart Lloyd's k-means [83]. Even though the case is indeed statistically proven in problems of lower dimensions, we propose that the claim does not hold in cases where the problems are of higher dimension.

Despite the slow convergence, we have discovered that PSO clustering does not guarantee cluster optimality in higher dimensional dataset. The algorithm suffers from suboptimal trajectory and degrading performance in higher dimension as a consequence of Section 3.3 which will be covered in detail in the following section.

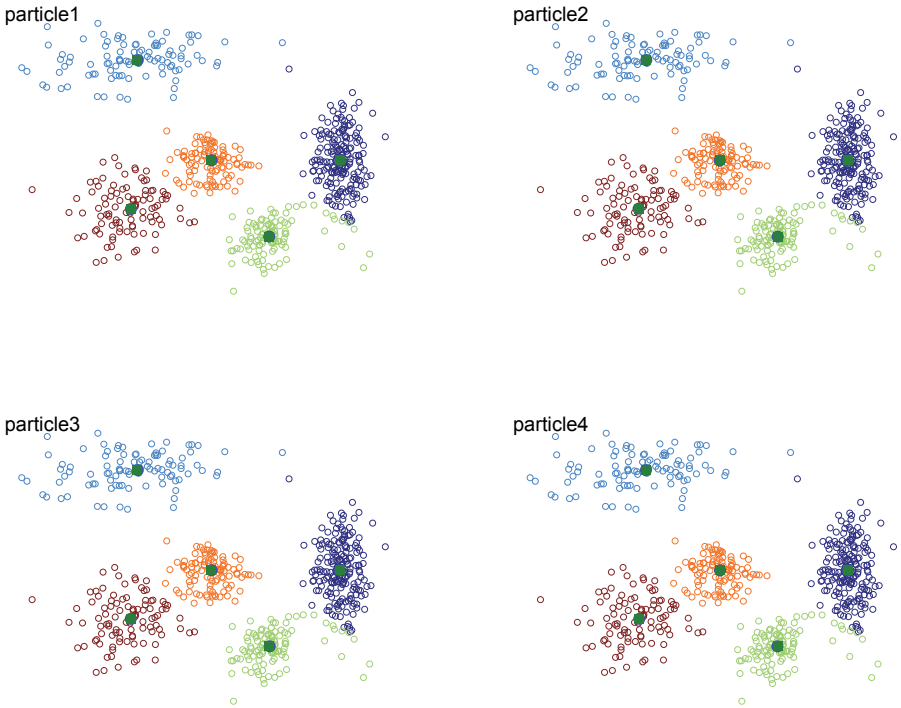
3.3 Suboptimal Convergence in Higher Dimension

Finding the optimum in prototype-based clustering problem is a rather delicate process. As has been proven back in the earlier chapter on divisive clustering, we know that the guaranteed optimality criterion of Equation 3.38 is achieved when all $\mathbf{z}$ is positioned at the mean of each cluster concurrently, such that $(\forall i), \mathbf{z}_i \leftarrow mean(\mathbb{C}_i)$. We discover a significant challenge in Van Der Merwe - Engelbrecht's paradigm when dealing with problems of higher dimension.

In Van Der Merwe - Engelbrecht's PSO Clustering, each update does not necessarily maximize the likelihood function. Unless at least one of the particles is pre-initialized near to the optimum, the probability of the swarm to converge to the maximum likelihood coordinates degrades exponentially as the number of classes or dimensions of the clustering problem increase.



(A) # function evaluation = 300



(B) # function evaluation = 1800; Converged

FIGURE 3.2: Initialization and Result: Van Der Merwe - Engelbrecht's PSO Clustering on a 2-Dimensional Dataset. Red square indicates swarm global best.

3.3.1 Trajectory Analysis

Consider an *dim*-dimensional K-classes clustering problem. For each particle, the objective function is as follows.

$$f(\mathbf{x}) = \frac{1}{K}\sum_{\mathbf{z}\in\mathbf{x}}\sum_{\mathbf{y}\in\mathbb{C}_z}\frac{1}{|\mathbb{C}_z|}d(\mathbf{y},\mathbf{z}), \tag{3.40}$$

$$= \frac{1}{K}\sum_{i=1}^{K}\sum_{\mathbf{y}\in\mathbb{C}_i}\frac{1}{|\mathbb{C}_i|}||\mathbf{y}-\mathbf{z}_i||^2, \tag{3.41}$$

$$= \frac{1}{K}\sum_{i=1}^{K}\sum_{\mathbf{y}\in\mathbb{C}_i}\sum_{j=1}^{dim}\frac{(y_j - z_{ij})^2}{|\mathbb{C}_i|}. \tag{3.42}$$

Taking the derivative of this equation with respect to $\mathbf{z}$, we can guarantee that $f(\mathbf{x}^*) = \arg\min_{\mathbf{x}} f(\mathbf{x})$ for the current cluster assignment when

$$\mathbf{z}_i^* = \frac{\sum_{\mathbf{y}\in\mathbb{C}_i}\mathbf{y}}{|\mathbb{C}_i|}, \tag{3.43}$$

where $\mathbf{x}^* = \{\mathbf{z}_1^*, \ldots, \mathbf{z}_K^*\}$. This formula is apparently the k-means update rule, $(\forall i), \mathbf{z}_i \leftarrow mean(\mathbb{C}_i)$. The likelihood function is maximized for each $\mathbf{z}_i$ when each $\mathbf{z}$ goes to the mean of each cluster in each dimension consecutively given the cluster assignment. However, the particles in PSO clustering are not aware of this information. Continuing, we can rewrite Equation 3.43 as follows,

$$\mathbf{z}_i^* = \mathbf{z}_i + \overbrace{\left(\frac{\sum_{\mathbf{y}\in\mathbb{C}_i}\mathbf{y}}{|\mathbb{C}_i|} - \mathbf{z}_i\right)}^{\delta_i}, \tag{3.44}$$

$$\{\mathbf{z}_1^*, \ldots, \mathbf{z}_K^*\} = \{\mathbf{z}_1 + \delta_1, \ldots, \mathbf{z}_K + \delta_K\} \tag{3.45}$$

$$\mathbf{x}^* = \mathbf{x}(t) + \Delta, \tag{3.46}$$

where $\Delta = \texttt{vectorize}\{\delta_1, \ldots, \delta_K\}$. Substituting to Equation 3.44 to Equation 3.42 we have,

$$f(\mathbf{x}^*) = \frac{1}{K}\sum_{i=1}^{K}\sum_{\mathbf{y}\in\mathbb{C}_i}\sum_{j=1}^{dim}\frac{(y_j - (z_{ij} + \delta_{ij}))^2}{|\mathbb{C}_i|} \tag{3.47}$$

Meanwhile the position of a particle at $t+1$ will be updated as follows,

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1), \tag{3.48}$$

$$\{\mathbf{z}_1(t+1), \ldots, \mathbf{z}_K(t+1)\} = \{\mathbf{z}_1 + \mathbf{v}_1, \ldots, \mathbf{z}_K + \mathbf{v}_K\} \tag{3.49}$$

The similarity between the two vectors, $\mathbf{v} = \texttt{vectorize}\{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ and $\Delta = \texttt{vectorize}\{\delta_1, \ldots, \delta_K\}$ can be calculated,

$$\cos(\Delta, \mathbf{v}) = \frac{\langle \Delta, \mathbf{v} \rangle}{|\Delta||\mathbf{v}|}. \tag{3.50}$$

The above equation is a sign matching problem: As long as every element in both vectors have the same sign, we can ensure that the angle between the two vectors is at most 90° such that $\cos(\Delta, \mathbf{v}) > 0$. Any particle whose trajectory satisfies this condition is guaranteed to minimize $f(\mathbf{x})$ and obtaining a better personal best.

The probability of any particle finding a better position along the vector of maximum likelihood would then be equivalent to the probability of having an angle lower than 90°, $p(\cos(\Delta, \mathbf{v}) > 0)$, which can be satisfied when $\langle \Delta, \mathbf{v} \rangle > 0$. For each particle, this probability can be calculated simply as the joint probability of each pair of elements in Δ and $\mathbf{v}$,

$$p(\langle \Delta, \mathbf{v} \rangle > 0) \cong \prod_{i=1}^{K} \prod_{j=1}^{dim} \overbrace{p(\delta_{ij} v_{ij} > 0)}^{<1}, \tag{3.51}$$

which approaches 0 at an exponential rate given large K or dim. Figure 3.3 gives an illustration of this phenomenon.

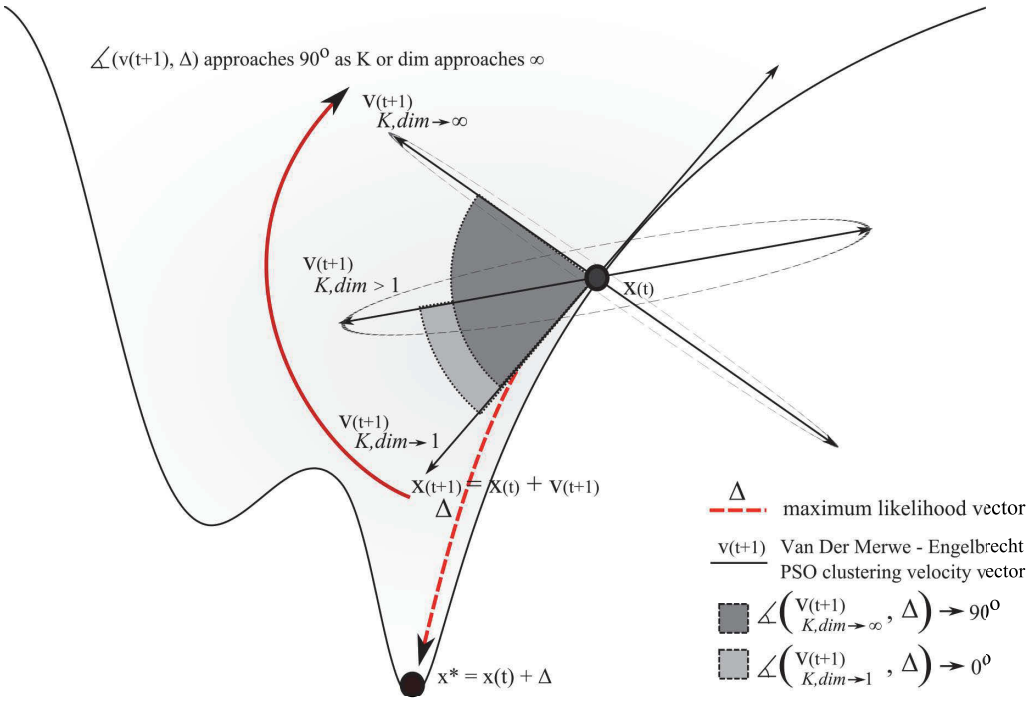


FIGURE 3.3: Trajectory of the Van Der Merwe - Engelbrecht PSO clustering vs K-means. $\measuredangle(v(t+1), \Delta) \to \pi/2$ as $K, dim \to \infty$. K = number of clusters, dim = dimension of the clustering problem.

3.3.2 Analogy Using Game Theory

Let us assume an all-knowing oracle appears to the player at the current time t. The oracle ensures the player that the velocity v^* leads to the maximum likelihood estimate given the current cluster assignment. The player disregards the oracle, and proceed with the velocity vector given by the swarm. Such situation is illustrated as follows.

Oracle: Let $\mathbf{z}_i^* \leftarrow mean(\mathbb{C}_i)$, then $\mathbf{z}_i^* = \mathbf{z}_i + \mathbf{v}_i^*$. The oracle gives the player $\mathbf{v}_i^*$.

Player (particle): disregards the oracle and proceed with $\hat{\mathbf{z}}_i = \mathbf{z}_i + \mathbf{v}_i$.

The cosine angle between $\mathbf{v}_i$ and $\mathbf{v}_i^*$ is

$$cos(\mathbf{v}_i, \mathbf{v}_i^*) = \frac{\langle \mathbf{v}_i \mathbf{v}_i^* \rangle}{|\mathbf{v}_i||\mathbf{v}_i^*|}. \tag{3.52}$$

The probability of the vector given by the swarm to align with the vector given by the oracle with at least 90 degree angle is then,

$$p(cos(\mathbf{v}_i, \mathbf{v}_i^*) > 0) = \overbrace{\prod_{d=1}^{dim} p(v_{id} v_{id}^* > 0)}^{\approx 0 \, when \, dim \to \infty}, \tag{3.53}$$

which converges to 0 at an exponential rate as the dimensionality of the clustering problem approaches infinity given that $p(v_{id} v_{id}^* > 0) < 1$.

Since a particle in the swarm assumes an oscillatory trajectory around its local best particle coordinate, the vector of the swarm is independent of that given by the oracle. Such characteristic would then give $p(v_{id} v_{id}^* > 0) < 1$ by definition.

3.3.3 Empirical Validation

A comparison between a randomly seeded PSO clustering and its deterministic predecessor on non-overlapping artificial gaussian datasets can be seen in Table 3.1. The dataset is generated

from five multivariate Gaussian with arbitrary dimensions as follows,

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 5 & 0.1 & 5 & 0.1 & \dots \\ 0.1 & 5 & 0.1 & 5 & \dots \\ 5 & 0.1 & 5 & 0.1 & \dots \\ 0.1 & 5 & 0.1 & 5 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 0 \\ 15 \\ 0 \\ 15 \\ \vdots \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 15.2 & 0.5 & 15.2 & 0.5 & \dots \\ 0.5 & 3.3 & 0.5 & 3.3 & \dots \\ 15.2 & 0.5 & 15.2 & 0.5 & \dots \\ 0.5 & 3.3 & 0.5 & 3.3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

$$\mu_3 = \begin{pmatrix} 10 \\ -3 \\ 10 \\ -3 \\ \vdots \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 1.6 & 0.5 & 1.6 & 0.5 & \dots \\ 0.5 & 2.8 & 0.5 & 2.8 & \dots \\ 1.6 & 0.5 & 1.6 & 0.5 & \dots \\ 0.5 & 2.8 & 0.5 & 2.8 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad \mu_4 = \begin{pmatrix} 6 \\ 5 \\ 6 \\ 5 \\ \vdots \end{pmatrix}, \Sigma_4 = \begin{pmatrix} 2.4 & -0.5 & 2.4 & -0.5 & \dots \\ -0.5 & 2.7 & -0.5 & 2.7 & \dots \\ 2.4 & -0.5 & 2.4 & -0.5 & \dots \\ -0.5 & 2.7 & -0.5 & 2.7 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

$$\mu_5 = \begin{pmatrix} 16 \\ 5 \\ 16 \\ 5 \\ \vdots \end{pmatrix}, \Sigma_5 = \begin{pmatrix} 1.4 & -0.5 & 1.4 & -0.5 & \dots \\ -0.5 & 10.1 & -0.5 & 10.1 & \dots \\ 1.4 & -0.5 & 1.4 & -0.5 & \dots \\ -0.5 & 10.1 & -0.5 & 10.1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{3.54}$$

The PSO was executed using the parameters described in Van Der Merwe and Engelbrecht's paper [83]. The number of particles is set to 30. Each algorithm were executed 100 times. One-tailed paired t-test were utilized to provide comparison on the total distortion after optimization against k-means. The distance was set to squared euclidean. The results can be seen in Table 3.1. A scatter plot comparing the performance of the algorithm with k-means and RCE^{r+} can be seen in Figure 3.4.

We can see that while PSO may exhibit comparable or better performance than k-means in clustering problems of lower dimensions / fewer clusters, it can be seen that the increase in the number of clusters and dimensionality of the clustering problem significantly degrades its overall performance. This result is to be expected, as has been theoretically proven in Section 3.3.

TABLE 3.1: Performance comparison between randomly seeded PSO clustering and k-means++

Dim	**K**	**PSO clustering**		**k-means++**		**p-value**
		# func. eval	total distortion	# func. eval	total distortion	
2	4	12000	8.1 ± 5.2	2.4 ± 3.0	7.9 ± 5.1	0.465
7	3	12000	24.8 ± 17.8	1.9 ± 2.3	15.2 ± 9.3	0.021
15	8	12000	226.9 ± 12.2	5.3 ± 5.3	30.9 ± 10.8	**<0.01**
30	8	12000	408.4 ± 15.2	6 ± 4	43.9 ± 15.3	**<0.01**

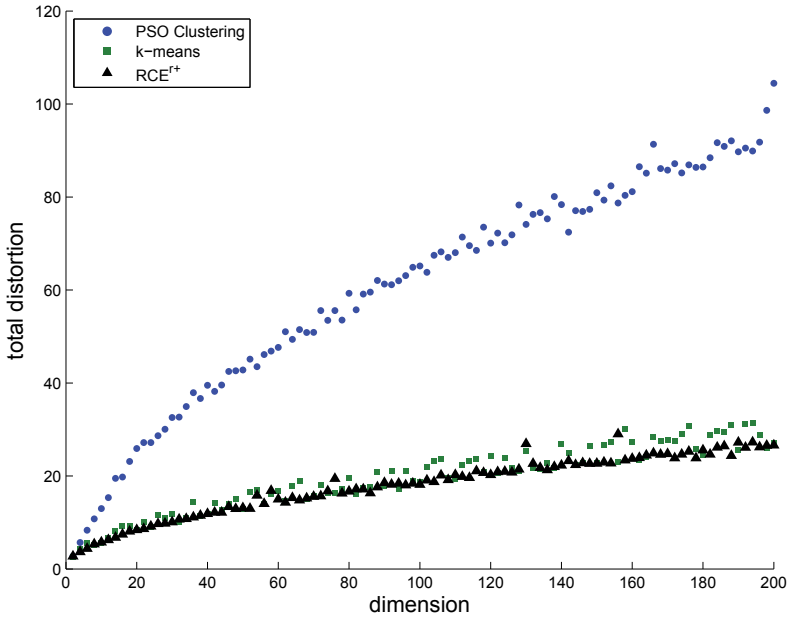


FIGURE 3.4: The Performance of PSO Clustering compared to k-means and RCE^{r+}. Distance: Squared Euclidean. PSO parameters: # particles = 30, maximum # function evaluations = 12000. RCE^{r+} parameters: # particles = 5, maximum # function evaluations = 100, substitution probability = 0.05, particle reset threshold = 15. $dim = \{2, \ldots, 200\}$, $k = 5$.

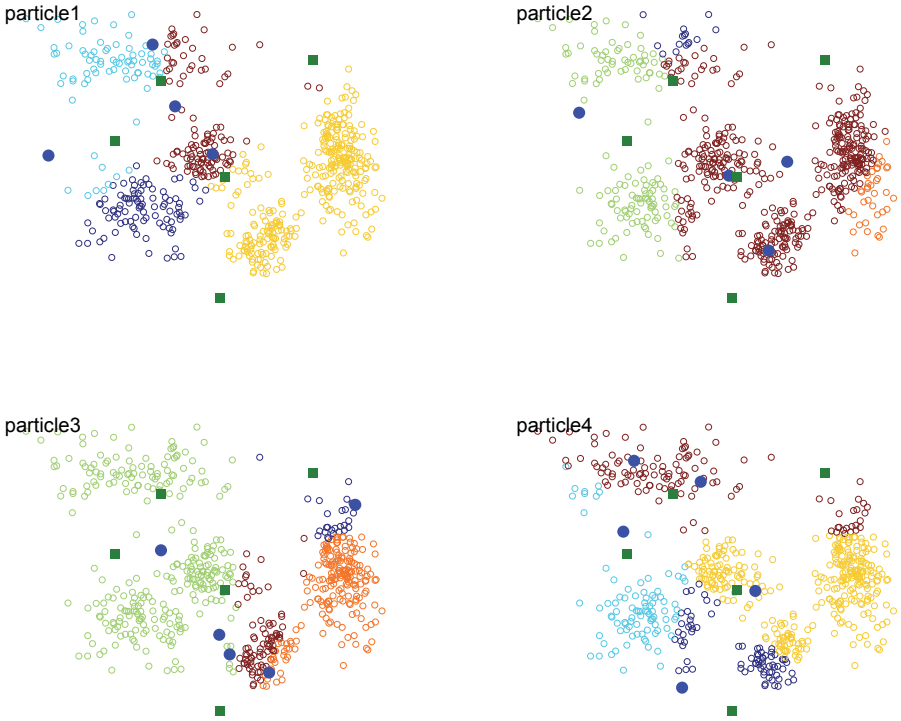
Figure 3.5 shows the trajectory of the particles on an 8-dimensional problem. The trajectory of the particles on the first two dimension are shown. This experiment validates both the theoretical and empirical proof of trajectory suboptimality of Van Der Merwe - Engelbrecht's PSO Clustering.

3.4 Summary

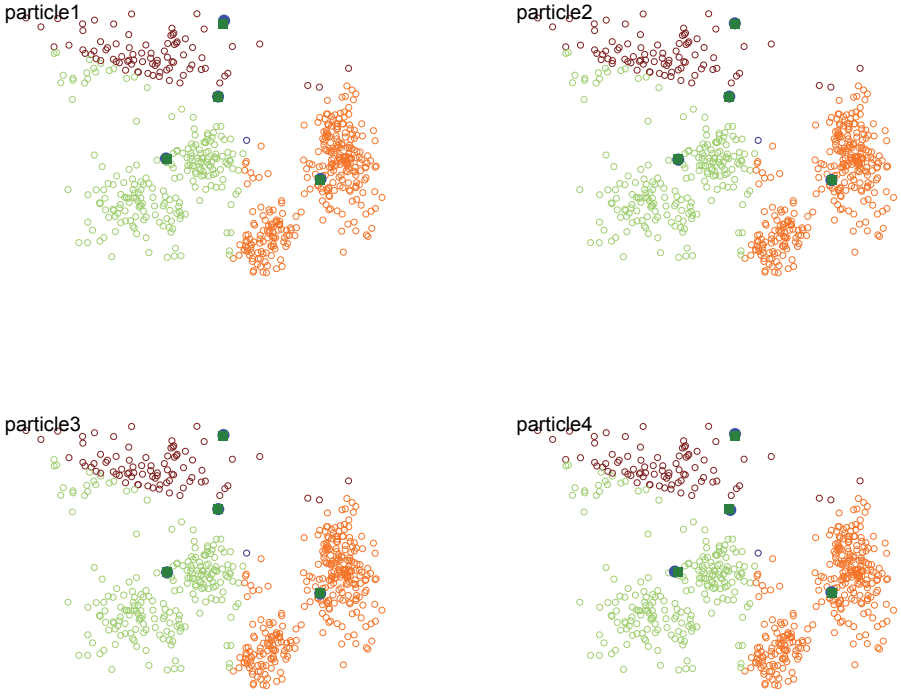
This chapter focuses on the general analysis on Particle Swarm Optimization (PSO) algorithm and PSO clustering. Several contributions has been made summarized as follows.

Lemma 3.1.1 reveals that the particle swarm optimization can be generally modeled as a second order linear system. Particularly, we discover that Jiang et al.'s stability boundary [81] is incorrectly derived.

In Section 3.3, significant issue in Van Der Merwe - Engelbrecht's proposition [83] is observed. Both theoretical and empirical validation reveal that *unless at least one of the particles is pre-initialized near to the optimum, the probability of the swarm to converge to the maximum likelihood coordinates degrades exponentially as the number of classes or dimensions of the clustering problem increase.*



(A) # function evaluation = 300



(B) # function evaluation = 1800; Converged

FIGURE 3.5: Trajectory of Van Der Merwe - Engelbrecht's PSO Clustering on an 8-Dimensional Dataset (4 times duplication of the two dimensions). Red square indicates swarm global best. The swarm converges to a severely suboptimal position.

Chapter 4

The Particle Swarm Clustering

Cohen and de Castro proposes an alternate view to clustering using particle swarm [75]. The proposal defines a rather unique perspective on the particle-data interaction within a swarm compared to Van Der Merwe - Engelbrecht's PSO Clustering. This chapter is dedicated to give an in depth discourse on the proposition.

The chapter is organized as follows. Section 4.1 defines the terminologies used in the Particle Swarm Clustering (PSC) family. Section 4.2 summarizes the algorithmic frameworks. Section 4.3 analyzes both computational and memory complexity of the PSC family. Section 4.4 provides an in depth discourse on the PSC particle stability and behavior. Section 4.5 presents the comparative empirical experiments against competing algorithms using a dataset openly available from the UCI machine-learning repository [84]. Finally Section 4.6 summarizes the contribution of this chapter.

4.1 Definitions

Definition 4.1.1 *(Swarm).* A swarm Θ represents a candidate partition of a dataset $\mathbb{Y} \in \mathbb{R}^{dim}$. The swarm consists of particles $\{\theta_1, \ldots, \theta_K\}$ and social memory $\{\mathbf{g}_1, \ldots, \mathbf{g}_N\} \in \mathbb{R}^{dim}$ as follows,

$$\Theta = \{\theta_1, \ldots, \theta_K; \mathbf{g}_1, \ldots, \mathbf{g}_N\}. \tag{4.1}$$

$N = |\mathbb{Y}|$ denotes the number of observations in $\mathbb{Y}$. The number of particles, $K = |\mathbb{C}|$, specifies the number of desired voronoi regions $\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_K\}$.

Definition 4.1.2 *(Particle).* A particle θ consists of a position vector $\mathbf{x} \in \mathbb{R}^{dim}$, a velocity vector $\mathbf{v} \in \mathbb{R}^{dim}$ and cognitive memory $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathbb{R}^{dim}$ as follows,

$$\theta = \{\mathbf{x}, \mathbf{v}; \mathbf{p}_1, \ldots, \mathbf{p}_N\}. \tag{4.2}$$

Each particle governs a voronoi region $\mathbb{C}_x$, with voronoi cell $\mathbf{x}$. Each data in $\mathbb{C}_x$ is crisply associated with the closest corresponding cell in the Euclidean space.

Definition 4.1.3 *(Position).* The position of a particle $\mathbf{x}$ denotes its literal location in the Euclidean space. $\mathbf{x}$ represents a potential prototype vector describing the location of a voronoi cell. The position of each particle is updated similarly to the standard PSO rule as follows,

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1), \tag{4.3}$$

where $\mathbf{v}$ denotes the velocity vector of the corresponding particle.

Definition 4.1.4 *(Cognitive Memory).* Each particle θ stores a cognitive memory $\mathbb{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathbb{R}^{dim}$. The cognitive memory stores the closest position of the corresponding particle in relation to each data vector in the dataset $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$. For each particle, the cognitive memory is stored in a $dim \times N$ matrix. Notice that as each particle is required to store such matrix, the $\mathbf{P}$ matrix of the swarm is a three-dimensional matrix with size of $dim \times N \times K$. The cognitive memory update rule is as follows,

$$\mathbf{p}_j = \begin{cases} \mathbf{x} & if \quad d(\mathbf{x}, \mathbf{y}_j) < d(\mathbf{p}_j, \mathbf{y}_j) \\ \mathbf{p}_j & otherwise \end{cases}, \tag{4.4}$$

where j denotes the index of data vectors, $d(\cdot, \cdot)$ denotes the distance between two vectors according to a pre-specified distance function.

Definition 4.1.5 *(Social Memory).* The swarm Θ stores the social memory $\mathbb{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_N\}$ which represents the position of the particle that has been closest to each data vector in the dataset $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$. The social memory can be expressed in a $dim \times N$ matrix format. The social memory update rule is as follows,

$$\mathbf{g}_j = \begin{cases} \mathbf{p}_{ij} & if \quad d(\mathbf{p}_{ij}, \mathbf{y}_j) < d(\mathbf{g}_j, \mathbf{y}_j) \\ \mathbf{g}_j & otherwise \end{cases}, \tag{4.5}$$

where i denotes the index of particles, j denotes the index of data vectors.

Definition 4.1.6 *(Winning Particle).* A winning particle θ_{win} is the particle which constituted voronoi region contains the most data compared to that of the rest of the particles in the swarm,

$$\theta_{win} = \arg\max_{\theta} |\mathbb{C}_{\theta}|. \tag{4.6}$$

Definition 4.1.7 *(Velocity).* The velocity vector $\mathbf{v}$ of a particle describes its movement trajectory in the Euclidean space. The velocity vector in Cohen - De Castro's PSC is updated based on the interaction of the current particle θ_i with respect to the data vector $\mathbf{y}_j$ as follows,

$$\mathbf{v}_{ij}(t+1) = \begin{cases} \omega \mathbf{v}_{ij}(t) + \alpha\varphi_{so} \circ \mathbf{so}_{ij}(t) + \beta\varphi_{sc} \circ \mathbf{sc}_{ij}(t) + \gamma\varphi_{co} \circ \mathbf{co}_{ij}(t) & if \quad \mathbb{C}_i \neq \emptyset \\ \omega \mathbf{v}_{ij}(t) + \varphi \circ (\mathbf{x}_{win} - \mathbf{x}_i) & otherwise \end{cases}, \tag{4.7}$$

where each $\varphi_{\bigcirc} \in \{0,1\} \in \mathbb{R}^{dim}$ denotes a uniform random vector, $\circ$ denotes Hadamard product, $\mathbf{so} \in \mathbb{R}^{dim}$ denotes the self-organizing vector, $\mathbf{sc} \in \mathbb{R}^{dim}$ denotes the social vector, $\mathbf{co} \in \mathbb{R}^{dim}$ denotes the cognitive vector, and $\mathbf{x}_{win}$ denotes the position of the winning particle θ_{win}. $\alpha, \beta,$ and $, \gamma$ are three user-specified constants which specifies the degree of magnitude of each term.

The velocity is upper and lower bounded by a maximum velocity bound, which is set to a percentage $\eta\%$ of the search space Ω, similarly to the general PSO to avoid swarm explosion as follows,

$$\mathbf{v}(t) = \max(\min(\mathbf{v}(t), \mathbf{v}_{\max}), -\mathbf{v}_{\max}), \tag{4.8}$$

$$\mathbf{v}_{max} = \eta\% \cdot \Omega. \tag{4.9}$$

As described compactly in Equation 4.7, it can be observed that the method for updating the velocity of a particle depends on two possible scenarios:

1. More than one data vector is closest to $\mathbf{x}_i$ such that the voronoi region consituted by $\mathbf{x}_i$ is not empty ($\mathbb{C}_i \neq \emptyset$). In this scenario, the particle will experience a force of attraction due to the data in $\mathbb{C}_i$.

2. There are no data vector which associated to $\mathbf{x}_i$ such that such that the voronoi region consituted by $\mathbf{x}_i$ is empty ($\mathbb{C}_i = \emptyset$). In this scenario, the particle will experience a force of attraction due to the winning particle θ_{win}.

Definition 4.1.8 *(Self-Organizing Vector).* The *self-organizing* vector $\mathbf{so}_{ij}$ describes the attraction vector imposed to a particle $\mathbf{x}_i$ due to the data $\mathbf{y}_j$ as follows,

$$\mathbf{so}_{ij} = \begin{cases} \mathbf{y}_j - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{4.10}$$

Definition 4.1.9 *(Social Vector).* The *social* vector $\mathbf{sc}_{ij}$ describes the attraction vector imposed to a particle $\mathbf{x}_i$ due to the social memory $\mathbf{g}_j$ associated with the data $\mathbf{y}_j$ as follows,

$$\mathbf{sc}_{ij} = \begin{cases} \mathbf{g}_j - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{4.11}$$

Definition 4.1.10 *(Cognitive Vector).* The *self-organizing* vector $\mathbf{co}_{ij}$ describes the attraction vector imposed to a particle $\mathbf{x}_i$ due to the cognitive memory $\mathbf{p}_{ij}$ of the corresponding particle $\mathbf{x}_i$ associated with the data $\mathbf{y}_j$ as follows,

$$\mathbf{co}_{ij} = \begin{cases} \mathbf{p}_{ij} - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{4.12}$$

4.2 Algorithmic Framework

4.2.1 Cohen - de Castro's Particle Swarm Clustering

Cohen and de Castro's original formulation is up to the extent described in Algorithm 4.15 [75]. The PSC does not compute any specific objective function or any indicator of cluster validity. While on one hand this formulation saves PSC a significant amount of computational burden associated to investigating the correctness of a particle movement, on the other hand such formulation generalizes the PSC to k-means when seen at a higher level.

4.2.2 Szabo's Modified PSC (mPSC)

The modified Particle Swarm Clustering (mPSC) was proposed by Szabo et al. in 2010 [74] in an attempt to reduce its computational complexity. The proposal suggests removing the inertia weight and velocity bound, effectively redefines the update rule (Algorithm 4.15 line 14) of the

Algorithm 4.15 Cohen - de Castro's Particle Swarm Clustering (PSC)

Input: A set of data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, Number of clusters K, search space $\Omega = [\mathbf{x}_{min} = \lfloor \mathbb{Y} \rfloor, \mathbf{x}_{max} = \lceil \mathbb{Y} \rceil]$, constants α, β, γ, velocity clamp $[\mathbf{v}_{min}, \mathbf{v}_{max}]$.
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$.

1: **for all** $\theta = \{\mathbf{x}, \mathbf{v}; \mathbf{p}_1, \ldots, \mathbf{p}_N\}$
2: $\quad \mathbf{x} \leftarrow rand(\Omega)$
3: $\quad \mathbf{v} \leftarrow \{0\}$
4: **end for**
5: **while** $t < t_{max}$
6: $\quad$ **for all** $\mathbf{y} \in \mathbb{Y}$
7: $\quad\quad I = \arg\min_{\mathbf{x}} d(\mathbf{y}, \mathbf{x})$
8: $\quad\quad$ **if** $d(\mathbf{x}_I, \mathbf{y}) < d(\mathbf{p}_{I,y}, \mathbf{y})$
9: $\quad\quad\quad \mathbf{p}_{I,y} \leftarrow \mathbf{x}_I$
10: $\quad\quad\quad$ **if** $d(\mathbf{p}_{I,y}, \mathbf{y}) < d(\mathbf{g}_y, \mathbf{y})$
11: $\quad\quad\quad\quad \mathbf{g}_y \leftarrow \mathbf{p}_{I,y}$
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end if**
14: $\quad\quad \mathbf{v}_I(t+1) \leftarrow \omega(t)\mathbf{v}_I(t) + \overbrace{\alpha\varphi_{so} \circ (\mathbf{y} - \mathbf{x}_I)}^{self-organizing} + \overbrace{\beta\varphi_{sc} \circ (\mathbf{g}_y - \mathbf{x}_I)}^{social} + \overbrace{\gamma\varphi_{co} \circ (\mathbf{p}_{I,y} - \mathbf{x}_I)}^{cognitive}$
15: $\quad\quad [\mathbf{v}_I(t+1)] = \max(\min(\mathbf{v}_I(t+1), \mathbf{v}_{\max}), -\mathbf{v}_{\max})$
16: $\quad\quad \mathbf{x}_I(t+1) \leftarrow \mathbf{x}_I(t) + [\mathbf{v}_I(t+1)]$
17: $\quad$ **end for**
18: $\quad$ **for all** $\mathbb{C}_\theta = \emptyset$
19: $\quad\quad \mathbf{v}_\theta(t+1) \leftarrow \omega(t)\mathbf{v}_\theta(t) + \varphi \circ (\mathbf{x}_{win} - \mathbf{x}_\theta)$
20: $\quad\quad [\mathbf{v}_\theta(t+1)] = \max(\min(\mathbf{v}_\theta(t+1), \mathbf{v}_{\max}), -\mathbf{v}_{\max})$
21: $\quad\quad \mathbf{x}_\theta(t+1) \leftarrow \mathbf{x}_\theta(t) + [\mathbf{v}_\theta(t+1)]$
22: $\quad$ **end for**
23: **end while**
24: **return** $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$

original PSC as follows,

$$\Delta\mathbf{x}_{ij}(t+1) = \begin{cases} \alpha\varphi_{so} \circ \mathbf{so}_{ij}(t) + \beta\varphi_{sc} \circ \mathbf{sc}_{ij}(t) + \gamma\varphi_{co} \circ \mathbf{co}_{ij}(t) & if \quad \mathbb{C}_i \neq \emptyset \\ \varphi \circ (\mathbf{x}_{win} - \mathbf{x}) & otherwise \end{cases}, \tag{4.13}$$

$$\mathbf{x}_{ij}(t+1) = \mathbf{x}_{ij}(t) + \Delta\mathbf{x}_{ij}(t+1). \tag{4.14}$$

Similarly to PSC, the mPSC remains true to the core principle of PSC where it does not incorporate any objective function to measure cluster quality. In fact the main difference of mPSC compared to its predecessor is the fact that the inertia weight is zeroed for all iterations $\omega = 0$, effectively detaching the velocity integrator from the transfer function.

The mPSC also generalizes to k-means when seen at a higher level. The algorithmic pseudocode of mPSC is shown in Algorithm 4.16. From the pseudocode, it is easily seen that overall complexity of the algorithm resembles that of the PSC, however the mPSC is slightly leaner due to the removal of a few min and max operand in Algorithm 4.15 lines 16 and 20.

Algorithm 4.16 Szabo's Modified Particle Swarm Clustering (mPSC)

Input: A set of data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, Number of clusters K, search space $\Omega = [\mathbf{x}_{min} = \lfloor \mathbb{Y} \rfloor, \mathbf{x}_{max} = \lceil \mathbb{Y} \rceil]$, constants α, β, γ.
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$.

for all $\theta = \{\mathbf{x}, \mathbf{v}; \mathbf{p}_1, \ldots, \mathbf{p}_N\}$
  $\mathbf{x} \leftarrow rand(\Omega)$
  $\mathbf{v} \leftarrow \{0\}$
end for
while $t < t_{max}$
  for all $\mathbf{y} \in \mathbb{Y}$
    $I = \arg\min_{\mathbf{x}} d(\mathbf{y}, \mathbf{x})$
    if $d(\mathbf{x}_I, \mathbf{y}) < d(\mathbf{p}_{I,y}, \mathbf{y})$
      $\mathbf{p}_{I,y} \leftarrow \mathbf{x}_I$
      if $d(\mathbf{p}_{I,y}, \mathbf{y}) < d(\mathbf{g}_y, \mathbf{y})$
        $\mathbf{g}_y \leftarrow \mathbf{p}_{I,y}$
      end if
    end if
    $\Delta\mathbf{x}_I(t+1) \leftarrow \overbrace{\alpha\varphi_{so} \circ (\mathbf{y} - \mathbf{x}_I)}^{self-organizing} + \overbrace{\beta\varphi_{sc} \circ (\mathbf{g}_y - \mathbf{x}_I)}^{social} + \overbrace{\gamma\varphi_{co} \circ (\mathbf{p}_{I,y} - \mathbf{x}_I)}^{cognitive}$
    $\mathbf{x}_I(t+1) \leftarrow \mathbf{x}_I(t) + \Delta\mathbf{x}_I(t+1)$
  end for
  for all $\mathbb{C}_\theta = \emptyset$
    $\Delta\mathbf{x}_\theta(t+1) \leftarrow \varphi \circ (\mathbf{x}_{win} - \mathbf{x}_\theta)$
    $\mathbf{x}_\theta(t+1) \leftarrow \mathbf{x}_\theta(t) + \Delta\mathbf{x}_\theta(t+1)$
  end for
end while
return $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$

4.2.3 Szabo's Fuzzy PSC (FPSC)

The Fuzzy PSC is simply a PSC with fuzzy membership instead of crisp membership. The fuzzification is done based on the distance between a data and its respective center using Dunn's fuzzy c-means principle as formulated in Equation 2.49. The rest of the concepts are similarly defined as Cohen - de Castro's PSC.

4.3 Complexity Analysis

4.3.1 Computational Complexity

The general computational complexity of the algorithms from the PSC family can be analyzed as follows. On each iteration of PSC computes of the distance between each data vector relative to each particle, and performing position updates for each data vector, imposing a total complexity of $\mathcal{O}(\delta + \kappa)$ on each iteration.

Algorithm 4.17 Szabo's Fuzzy Particle Swarm Clustering (FPSC)

Input: A set of data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, Number of clusters K, fuzzification parameter m, search space $\Omega = [\mathbf{x}_{min} = \lfloor \mathbb{Y} \rfloor, \mathbf{x}_{max} = \lceil \mathbb{Y} \rceil]$, constants α, β, γ, velocity clamp $[\mathbf{v}_{min}, \mathbf{v}_{max}]$.
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$.

for all $\theta = \{\mathbf{x}, \mathbf{v}; \mathbf{p}_1, \ldots, \mathbf{p}_N\}$
  $\mathbf{x} \leftarrow rand(\Omega)$
  $\mathbf{v} \leftarrow \{0\}$
end for
$(\forall \mathbf{x}, \mathbf{y}), u(\mathbf{x}, \mathbf{y}) = \dfrac{d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}{\sum_{i=1}^{K} d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}$,
while $t < t_{max}$
  for all $\mathbf{y} \in \mathbb{Y}$
    $I = \arg\max_{\mathbf{x}} u(\mathbf{x}, \mathbf{y})$
    if $u(\mathbf{x}_I, \mathbf{y}) > u(\mathbf{p}_{I,y}, \mathbf{y})$
      $\mathbf{p}_{I,y} \leftarrow \mathbf{x}_I$
      if $u(\mathbf{p}_{I,y}, \mathbf{y}) > u(\mathbf{g}_y, \mathbf{y})$
        $\mathbf{g}_y \leftarrow \mathbf{p}_{I,y}$
      end if
    end if
    $\mathbf{v}_I(t+1) \leftarrow \omega(t)\mathbf{v}_I(t) + \overbrace{\alpha\varphi_{so} \circ (\mathbf{y} - \mathbf{x}_I)}^{self-organizing} + \overbrace{\beta\varphi_{sc} \circ (\mathbf{g}_y - \mathbf{x}_I)}^{social} + \overbrace{\gamma\varphi_{co} \circ (\mathbf{p}_{I,y} - \mathbf{x}_I)}^{cognitive}$
    $[\mathbf{v}_I(t+1)] = \max(\min(\mathbf{v}_I(t+1), \mathbf{v}_{\max}), -\mathbf{v}_{\max})$
    $\mathbf{x}_I(t+1) \leftarrow \mathbf{x}_I(t) + [\mathbf{v}_I(t+1)]$
    $(\forall \mathbf{x}), u(\mathbf{x}, \mathbf{y}) = \dfrac{d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}{\sum_{i=1}^{K} d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}$,
  end for
  for all $\mathbb{C}_\theta = \emptyset$
    $\mathbf{v}_\theta(t+1) \leftarrow \omega(t)\mathbf{v}_\theta(t) + \varphi \circ (\mathbf{x}_{win} - \mathbf{x}_\theta)$
    $[\mathbf{v}_\theta(t+1)] = \max(\min(\mathbf{v}_\theta(t+1), \mathbf{v}_{\max}), -\mathbf{v}_{\max})$
    $\mathbf{x}_\theta(t+1) \leftarrow \mathbf{x}_\theta(t) + [\mathbf{v}_\theta(t+1)]$
    $(\forall \mathbf{x}), u(\mathbf{x}, \mathbf{y}) = \dfrac{d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}{\sum_{i=1}^{K} d(\mathbf{x}, \mathbf{y})^{-\frac{1}{m-1}}}$,
  end for
  if $\sum_{i=1}^{K}\sum_{j=1}^{N} u_{ij}^m ||\mathbf{y}_j - \mathbf{x}_i||^2$ does not improve
    break;
  end if
end while
return $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$

δ denotes the total cost of calculating the distance function (including communication overhead) and updating the cluster membership of each data vector relative to each particle position vector.

κ denotes the cost of updating the position of each particle (consisting of random number generation, position update, and memory matrices update). As a general overview, on each iteration the PSC needs to generate $3 \times N \times dim$ random numbers, and also perform N position updates in the dim-dimensional space.

The worst case complexity of the PSC algorithm variants for an iteration are,

$$\delta_{PSC} = \delta_{mPSC} = \delta_{FPSC} = \overbrace{\mathcal{O}(NKdim\mathfrak{D})}^{\text{pairwise distance calculation}}, \tag{4.15}$$

where $\mathfrak{D}$ denotes the cost of calculating the distance between two vectors, and

$$\kappa_{PSC} = \mathcal{O}(\overbrace{19Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}}), \tag{4.16}$$

$$\kappa_{mPSC} = \mathcal{O}(\overbrace{16Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}}), \tag{4.17}$$

$$\kappa_{FPSC} = \mathcal{O}(\overbrace{19Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}} + \overbrace{NK\mathfrak{F}}^{\text{fuzzification}}), \tag{4.18}$$

where $\mathfrak{F}$ is the cost of calculating floating point powers and divisions for a fractional number. The overall complexity of the algorithm is then

$$\mathcal{O}(T(\delta + \kappa)) = T \times \left(\begin{array}{l} \mathcal{O}(NKdim\mathfrak{D}) + \mathcal{O}(\log_2 3Ndim) + \cdots \\ \quad \begin{cases} \mathcal{O}(18Ndim + NK + 4N) & if\ PSC \\ \mathcal{O}(15Ndim + NK + 4N) & if\ mPSC \\ \mathcal{O}(18Ndim + NK + 4N + NK\mathfrak{F}) & if\ FPSC \end{cases} \end{array} \right), \tag{4.19}$$

where T denotes the number of iterations.

4.3.1.1 Computational Complexity of δ

The total cost of calling the distance and membership update function is specified by δ. Naïvely, the theoretical worst case computational complexity for the *delta* calculation is

$$\delta_{PSC} = \delta_{mPSC} = \delta_{FPSC} = \mathcal{O}(NKdim\mathfrak{D}), \tag{4.20}$$

where $\mathfrak{D}$ denotes the cost of calculating the distance between two vectors. Optimizing δ for PSC is difficult due to the serial dependency between position update and distance calculation. This particular 'for each data vector' loop in the PSC formulation creates a significant bottleneck to the overall performance of the algorithm that restricts its scalability.

Programmatically, minimizing the usage of this for loop can be done elegantly using batch matrix operation for all observations and particles and making use of vectorization and parallel processing.

There exists algorithmic variants (e.g. Coppersmith – Winograd algorithm [160], Williams algorithm [161]) for efficient calculation of matrix multiplication (e.g. two $n \times n$ matrices in $\mathcal{O}(n^{\approx 2.373})$ time) [160, 161]. However implementing this approach in PSC would require major alteration in its algorithmic architecture [4, 71, 73]. In its original formulation, each time a data vector is presented to the PSC swarm, one of the particle has to move as a consequence, and the pre-calculated distance matrix has to be naïvely recalculated as it would be obsolete each time a movement is made.

To empirically observe the effect of δ, we compare two scenarios as follows. The first scenario calls the distance matrix calculation function using the PSC paradigm: naïve for loop over all data vectors. The second scenario uses a pre-optimized batch matrix computation. For this particular simulation, Matlab is proper framework due to the fact that it is a specifically optimized framework for matrix operations. The simulations use a file `distmat.m`, a fully vectorized distance matrix calculation function which takes three inputs: the first observation matrix; the centroid matrix; and an index for specifying the distance function to be used (e.g. 2 = "squared Euclidean distance").

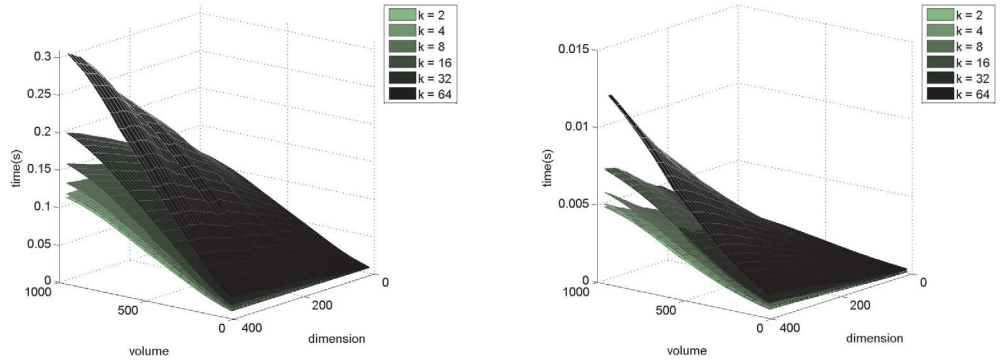
Scenario I: Naïve For Loop. The first scenario calls `distmat.m` using the PSC paradigm: naïve for loop over all data vectors. For each dimension and volume, the iteration is repeated 100 times to extract the central tendency of the time complexity for the particular dimension/volume. The Matlab code is as follows.

```
1 for j = 1:size(y,2) % loop over all data vectors
2     % invoke call to the squared euclidean distance function
3     distmat(y(:,j),x,2);
4 end
```

Scenario II: Matrix Operation. The second scenario executes the function `distmat.m` function using pre-optimized batch matrix operations. For each dimension and volume, the iteration is repeated 100 times to extract the central tendency for the particular dimension/volume. The Matlab code is as follows.

```
1 % Calculate the pairwise distance between y and x
2 distmat(y,x,2);
```

The resulting mesh plot of the time matrices with increasing number of voronoi regions $k = \{2, 4, 8, 16, 32, 64\}$ can be seen in Figure 4.1. The machine used in this simulation is a laptop computer with Intel Core i5 M520 @ 2.4 Ghz, 4 GB of RAM, running Windows 7.



(A) Time complexity of the distance computation using for loop.

(B) Time complexity of the distance computation optimized using matrix operation.

FIGURE 4.1: Time complexity of the distance computation δ.

As shown in Figure 4.1a, the inevitable usage of for loops poses a significant bottleneck that restricts the PSC to scale to larger datasets, both in term of volume and dimension. It can be seen that with K = 64, dim = 400, and N = 1000, a PSC iteration requires 0.26 seconds on average on the simulation machine.

`distmat.m` has been designed to utilize proper usage of efficient matrix operations and vectorization, hence minimizing the use of unnecessary for loops. It can be seen in Figure 4.1b that with K = 64, dim = 400, and N = 1000, a batch matrix computation requires 0.011 seconds on average using the machine, which is 23.6364 times quicker than an iteration of the PSC using the same parameters (0.26 seconds).

4.3.1.2 Computational Complexity of κ

The computational complexity of particle update κ is the second contributor to the PSC complexity. The main contributing factors can be summarized as follows:

1. The complexity of the random number generation which requires at most $3\times(N+K-1)\times dim$ pseudorandom floats or $\mathcal{O}(\log_2(3(N+K-1)dim))$ each iteration using Mersenne Twister pseudorandom number generator [162];

2. The complexity of velocity and position update which involves $7\times(N+K-1)\times dim$ multiplications for PSC (or $6\times(N+K-1)\times dim$ for mPSC), $5\times(N+K-1)\times dim$ additions, $2\times(N+K-1)\times dim$ logical operations for velocity clamp, and $2\times(N+K-1)\times dim$ assignments totaling up to $\mathcal{O}(16(N+K-1)dim)$ (or $\mathcal{O}(13(N+K-1)dim)$ for mPSC) each iteration;

3. The complexity of choosing the winning particle $\mathcal{O}(NK)$ for each data and updating the memory matrix which involves at most $2 \times N$ logical operations and $2 \times N + 2 \times N \times dim$ assignments, yielding an overall complexity of $(\mathcal{O}(NK + 4N + 2Ndim))$ each iteration, and;

4. Specifically for FPSC, fuzzification of the pairwise distances which poses an additional $\mathcal{O}((N+K-1)K)$ complexity per iteration.

For most clustering problems it is usually safe to assume the number of cluster K to be less than the number of data N such that $N + K - 1 \approx N$. Incorporating this simplification, the theoretical worst case complexities of the algorithms of the PSC family can then be summarized as follows,

$$\kappa_{PSC} = \mathcal{O}(\overbrace{16Ndim + \log_2 3Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}}), \tag{4.21}$$

$$\kappa_{mPSC} = \mathcal{O}(\overbrace{13Ndim + \log_2 3Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}}), \tag{4.22}$$

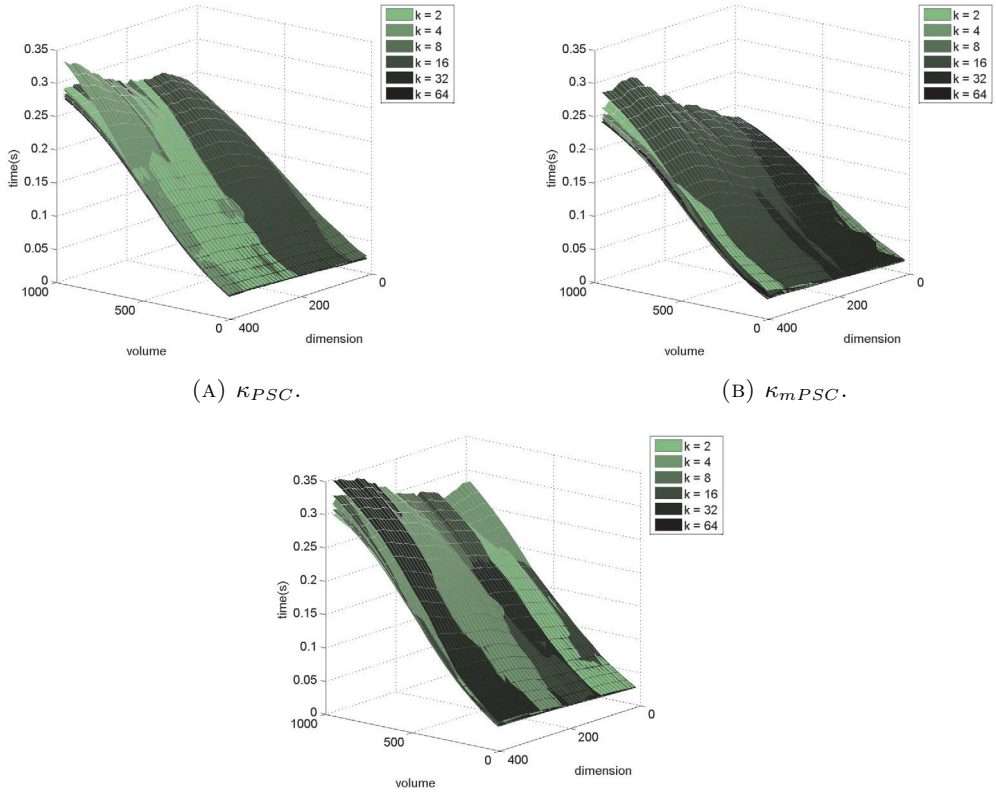
$$\kappa_{FPSC} = \mathcal{O}(\overbrace{16Ndim + \log_2 3Ndim}^{\text{position update}} + \overbrace{NK + 4N + 2Ndim}^{\text{memory matrix update}} + \overbrace{NK\mathfrak{F}}^{\text{fuzzification}}), \tag{4.23}$$

where $\mathfrak{F}$ is the cost of calculating floating point powers and divisions for a fractional number which can be rather expensive.

In order to empirically validate the theoretical computational complexity we coded a simulation testbench to investigate the time complexity of the PSC, mPSC and FPSC update mechanisms as described in Algorithm 4.15, Algorithm 4.16, and Algorithm 4.17, respectively.

The machine used in this simulation is a laptop computer with Intel Core i5 M520 @ 2.4 Ghz, 4 GB of RAM, running Windows 7. The overall result of the simulation for $k = \{2, 4, 8, 16, 32, 64\}$ can be seen in Figure 4.2.

As can be seen particularly in Figure 4.2, the mPSC, shown in Figure 4.2b, is the leanest among the PSC variants (0.24 seconds per iteration at N = 1000, dim = 400). FPSC exhibits the highest computational complexity (0.33 seconds per iteration at N = 1000, dim = 400), shown in Figure 4.2c. The update complexity of PSC, shown in Figure 4.2a, sits in the middle of the two (0.28 seconds per iteration at N = 1000, dim = 400). Unlike δ, we see that κ is minimally affected by the number of voronoi regions, K. This observation is consistent with the theoretical analysis in Equation 4.21, Equation 4.22, and Equation 4.23.



(A) κ_{PSC}.

(B) κ_{mPSC}.

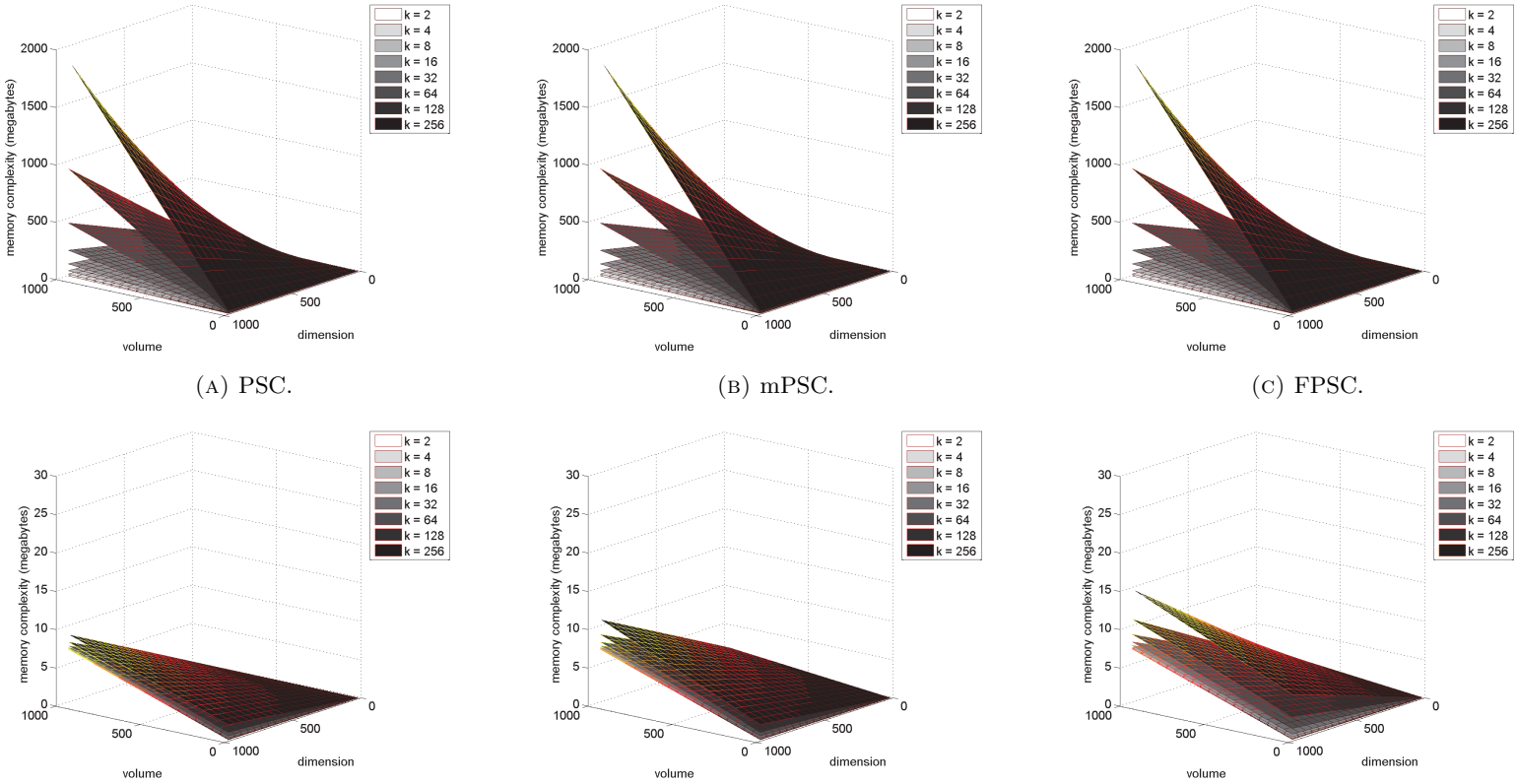
(C) κ_{FPSC}.

FIGURE 4.2: Time complexity of position update κ of PSC, mPSC, and FPSC with varying K.

4.3.2 Memory Complexity

The breakdown of the memory complexity of the PSC algorithm families and other clustering algorithms can be seen in Table 4.1. The estimated total memory complexity is computed by summing all the applicable complexities for the particular algorithm. An experiment is carried in Matlab to observe the memory requirement of each algorithm during runtime. The results are presented in Figure 4.3.

As can be seen in Figure 4.3 and Table 4.1, the algorithms of the PSC families are considerably expensive in terms of memory complexity. Our benchmark test revealed that in order to cluster 1000 observations of 1000 dimensional double precision data, the PSC families require as much as 2GB memory; whereas k-means, fuzzy c-means, and RCE^{r+} 2014 require memory allocation of lower than 15MB. This relatively high memory requirement is a significant scalability bottleneck when dealing with larger, higher dimensional datasets.



(A) PSC. (B) mPSC. (C) FPSC.

(D) k-means. (E) Fuzzy c-means. (F) RCE^{r+} (2014).

FIGURE 4.3: Memory complexity of various algorithms when clustering double precision floating point numbers. The runtime test is carried using Matlab. It can be easily observed that the PSC families are quite expensive in terms of its memory requirements.

TABLE 4.1: Memory complexity of the PSC families vs other clustering algorithms

Entity	Complexity	Fuzzy c-means	k-means	PSC [75]	mPSC [74]	FPSC [70]	RCE^{r+} (2012) [4]	RCE^{r+} (2014) [55]	Description
$\mathbf{Y}$	$\mathcal{O}(Ndim)$	✓	✓	✓	✓	✓	✓	✓	The data vectors: $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$
$\mathbf{X}$	$\mathcal{O}(Kdim)$	✓	✓	✓	✓	✓	✓	✓	The particle position vectors: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$
$\mathbf{V}$	$\mathcal{O}(Kdim)$			✓		✓	✓	✓	The particle velocity vectors: $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\} \in \mathbb{R}^{dim}$
$\mathbf{v}_{max}$	$\mathcal{O}(dim)$			✓		✓	✓	✓	The maximum velocity vector: $\mathbf{v}_{max} \in \mathbb{R}^{dim}$
U	$\mathcal{O}(N)$		✓	✓	✓		✓	✓	The crisp label vector or binary indicator matrix: $\mathsf{U} = \{\mathsf{u}_1, \ldots, \mathsf{u}_N\} \in \mathbb{R}^1$
$\mathbf{U}$	$\mathcal{O}(NK)$	✓				✓		✓	The fuzzy membership matrix: $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\} \in \mathbb{R}^K$
$\mathbf{X}_{best}$	$\mathcal{O}(Kdim)$						✓	✓	The best position matrix: $\mathbf{X}_{best} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}_{best} \in \mathbb{R}^{dim}$
$f(\mathbb{C}_{Xbest}, \mathbf{Y})$	$\mathcal{O}(1)$						✓	✓	The quality of the voronoi tessellation imposed by $\mathbf{X}_{best}$
$\mathbf{G}$	$\mathcal{O}(Ndim)$			✓	✓	✓	✓		The swarm social memory vector for each data: $\mathbf{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_N\} \in \mathbb{R}^{dim}$
$d(\mathbf{g}_j, \mathbf{y}_j)_{\{\forall i, \forall j\}}$	$\mathcal{O}(N)$			✓	✓	✓	✓		The distance between the vectors in the social memory relative to its corresponding data vector, $d(\mathbf{p}_{i,j}, \mathbf{y}_j) \in \mathbb{R}^1$
$\mathbf{P}_{\{1,\ldots,K\}}$	$\mathcal{O}(NKdim)$			✓	✓	✓	✓		The cognitive memory for each particle, for each data: $\mathbf{P}_i = \{\mathbf{p}_{i,1}, \ldots, \mathbf{p}_{i,N}\} \in \mathbb{R}^{dim}$
$d(\mathbf{p}_{i,j}, \mathbf{y}_j)_{\{\forall i, \forall j\}}$	$\mathcal{O}(NK)$			✓	✓	✓	✓		For each particle, the relative distance between the cognitive memory vector and its corresponding data vector, $d(\mathbf{p}_{i,j}, \mathbf{y}_j) \in \mathbb{R}^1$

4.4 Trajectory Analysis

4.4.1 Stability and Convergence

Theorem 4.1 (PSC's resemblance to k-means)**.** *Under the condition where α, β, γ, and ω obeys the stability constraints,*

$$0 \leq \omega < 1, \text{ and } 0 < \alpha + \beta + \gamma < \frac{2 + 2\omega}{|\mathbb{C}_x|}, \tag{4.24}$$

each particle would converge to

$$x(t \to \infty) = \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} \frac{\alpha y + \beta p_{xy}(t \to \infty) + \gamma g_y(t \to \infty)}{\alpha + \beta + \gamma}. \tag{4.25}$$

As a result, a monotonic decrease in the distortion function is guaranteed if and only if α *is non-zero regardless of zero values in* β *and* γ*, effectively reducing the PSC to k-means,*

$$x(t \to \infty) = \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y, \tag{4.26}$$

which implies the sensitivity of PSC to initialization, similarly with that of k-means.

Proof: Seen from a specific dimension, the PSC velocity update rule for the particle θ due to the data vector $\mathbf{y} \in \mathbb{C}_x$ is as follows,

$$\begin{aligned} v(t+1) &= \omega(t)v(t) + \alpha\varphi_{so}(y - x(t)) + \beta\varphi_{sc}(g - x(t)) + \gamma\varphi_{co}(p - x(t)), \\ &= \omega v(t) - (\alpha\varphi_{so} + \beta\varphi_{sc} + \gamma\varphi_{co})x(t) + \alpha\varphi_{so}y + \beta\varphi_{sc}g(t) + \gamma\varphi_{co}p(t). \\ x(t+1) &= x(t) + v(t+1), \\ &= \omega v(t) + \big(1 - (\alpha\varphi_{so} + \beta\varphi_{sc} + \gamma\varphi_{co})\big)x(t) + \alpha\varphi_{so}y + \beta\varphi_{sc}g(t) + \gamma\varphi_{co}p(t). \end{aligned} \tag{4.27}$$

The uniform random variables $\varphi_{\bigcirc} \in \{0, 1\}$ can be represented by its supremum $\sup(\varphi_{\bigcirc}) = 1$. Let $\mathbf{X}(t) = [x(t), v(t)]^T$ and $\mathbf{U}(t) = [y, p(t), g(t)]^T$, the PSC can expressed in an explicit discrete time-invariant state space format,

$$\begin{bmatrix} E[x(t+1)] \\ E[v(t+1)] \end{bmatrix} = \overbrace{\begin{bmatrix} 1 - (\alpha + \beta + \gamma) & \omega \\ -(\alpha + \beta + \gamma) & \omega \end{bmatrix}}^{\sup(\mathbf{A})} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \overbrace{\begin{bmatrix} \alpha & \beta & \gamma \\ \alpha & \beta & \gamma \end{bmatrix}}^{\sup(\mathbf{B})} \begin{bmatrix} y \\ p(t) \\ g(t) \end{bmatrix} \tag{4.28}$$

$$\mathbf{Y}(t) = \overbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}^{\mathbf{C}} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}. \tag{4.29}$$

The transfer function is consequently,

$$\mathbf{H}(z) = \frac{1}{z^2 + (\alpha + \beta + \gamma - \omega - 1)\, z + \omega} \begin{bmatrix} \alpha z & \beta z & \gamma z \end{bmatrix}, \tag{4.30}$$

Which is the same second order system in Lemma 3.1.1. Using the same approach we have a guaranteed convergence due to the data $y \in \mathbb{C}_x$ when

$$0 \leq \omega < 1, \text{ and } 0 < \alpha + \beta + \gamma < 2 + 2\omega. \tag{4.31}$$

With the same spirit as Equation 3.37, without loss of generality, the inputs to the system (self-organizing, cognitive and social positions) can be represented as three step functions with gains of Y, P and G. Assuming stability, applying final value theorem to the PSC transfer function yields,

$$\begin{aligned} \lim_{t\to\infty} x(t) &= \lim_{z\to 1}\{(z-1)\mathbf{H}(z)\mathbf{U}(z)\}, \\ &= \frac{\alpha Y + \beta P + \gamma G}{\alpha + \beta + \gamma}, \end{aligned} \tag{4.32}$$

which holds for all $y \in \mathbb{C}_x$, such that,

$$\begin{aligned} \lim_{t\to\infty} x_{i1}(t) &= \frac{\alpha y_1 + \beta p_{i1}(t\to\infty) + \gamma g_1(t\to\infty)}{\alpha + \beta + \gamma} \\ \lim_{t\to\infty} x_{i2}(t) &= \frac{\alpha y_2 + \beta p_{i2}(t\to\infty) + \gamma g_2(t\to\infty)}{\alpha + \beta + \gamma} \\ &\vdots \\ \lim_{t\to\infty} x_{ij}(t) &= \frac{\alpha y_j + \beta p_{ij}(t\to\infty) + \gamma g_j(t\to\infty)}{\alpha + \beta + \gamma} \end{aligned} \tag{4.33}$$

where x_{ij} denotes the expected position of the i^{th} particle due to the j^{th} data vector. The resultant vector is therefore

$$\begin{aligned} \lim_{t\to\infty} x_i(t) &= \lim_{t\to\infty} \sum_{j=1}^{|\mathbb{C}_{x_i}|} x_{ij}(t), \\ &= \sum_{j=1}^{|\mathbb{C}_x|} \frac{\alpha y_j + \beta p_{ij}(t\to\infty) + \gamma g_j(t\to\infty)}{\alpha + \beta + \gamma}. \end{aligned} \tag{4.34}$$

This characteristic is problematic since it implies that $(\forall i), x_i$ or can be described simply as $x \in \mathbb{X}$ will continue to bounce towards each $y \in \mathbb{C}_x$ in a cyclical pattern unless the relative effect of each term — self-organizing; social; and cognitive — is made sufficiently small such that the resultant vector approximates the maximum likelihood estimate: the coordinate of the voronoi cell.

This constraint obtained from the resultant consequently imposes an important criterion to enable each particle to converge. One way to achieve such condition is by scaling α, β and γ proportionally to the cardinality of the corresponding voronoi region $|\mathbb{C}_x|$: the number of data vectors in the cluster.

This proposition makes intuitive sense when the outer loop (**for all y** $\in \mathbb{Y}$) is treated as a function of a continuous time system. Each of this loop constitutes an iteration where the velocity of a particle x is updated $|\mathbb{C}_x|$ times: a proportion of the volume of the total data $|\mathbb{C}_x| = \eta\%|\mathbb{Y}|$. According to control theory, this discretization phenomenon can be treated as the system being sampled with the sampling frequency $f_s = |\mathbb{C}_x|$ as follows,

$$\mathbf{H}(z) = T\mathbf{H}_{\mathbb{C}x}(z), T = \frac{1}{f_s}, f_s = |\mathbb{C}_x|, \tag{4.35}$$

$$= \frac{1}{|\mathbb{C}_x|} \frac{1}{z^2 + \Big(\check{\alpha} + \check{\beta} + \check{\gamma} - \omega - 1\Big) z + \omega} \begin{bmatrix} \check{\alpha}z & \check{\beta}z & \check{\gamma}z \end{bmatrix}. \tag{4.36}$$

Consequently, the effect of the poles to the discrete transfer function $\mathbf{H}_{\mathbb{C}x}(z)$ is obviously magnified linearly by a gain of $|\mathbb{C}_x|$. Hence incorporating $|\mathbb{C}_x|$ to the feasible convergence bounds yields the proper scaling as follows

$$0 \leq \omega < 1, \text{ and } 0 < \check{\alpha} + \check{\beta} + \check{\gamma} < \frac{2 + 2\omega}{|\mathbb{C}_x|}. \tag{4.37}$$

The resultant using this updated bounds reflects convergence towards the center of $\mathbb{C}_x$,

$$x(t \to \infty) = \sum_{y \in \mathbb{C}_x} \frac{\check{\alpha}y + \check{\beta}p_{xy}(t \to \infty) + \check{\gamma}g_y(t \to \infty)}{\check{\alpha} + \check{\beta} + \check{\gamma}}, \tag{4.38}$$

$$= \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} \frac{\alpha y + \beta p_{xy}(t \to \infty) + \gamma g_y(t \to \infty)}{\alpha + \beta + \gamma}. \qquad \square \tag{4.39}$$

The convergence of this equation depends on whether the value of α is zero/nonzero.

Case 1: $\alpha > 0$

Note that that $(\forall y), \{p_{xy}(0), g_y(0)\}$ are initialized as $x(0)$ which is the very first particle that the data sees as it assigns itself to $\mathbb{C}_x$. Notice that on every iteration the value of the self-organizing vector is

$$so_y(t) = \frac{\alpha}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} (y - x(t)), \tag{4.40}$$

which guarantees a consistent decrease in distortion with respect to the cluster center such that,

$$0 < d\left(x(t+1), \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y\right) < d\left(x(t), \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y\right) < \ldots < d\left(x(0), \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y\right). \tag{4.41}$$

Note that the swarm will store $x(t)$ into the both memories when a decrease in distortion on an observation y is detected. Consequently both memories will then closely follow $x(t)$ such that $p_{xy}(t) \to x(t)$ and $g_y(t) \to x(t)$. Under the stability bounds in Equation 4.39 it is easy to see that both the memories for all y tend to converge to its corresponding cluster center,

$$(y \in \mathbb{C}_x), p_{xy}(t \to \infty) = g_y(t \to \infty) \to \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y. \tag{4.42}$$

Under this condition, Equation 4.39 converges to

$$x(t \to \infty) = \frac{1}{|\mathbb{C}_x|(\alpha + \beta + \gamma)} \left(\sum_{y \in \mathbb{C}_x} \alpha y + \sum_{y \in \mathbb{C}_x} \beta y + \sum_{y \in \mathbb{C}_x} \gamma y \right). \tag{4.43}$$

$$= \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y, \tag{4.44}$$

which is the k-means update formula. ■

Based on this analysis, β and γ can be ignored as the values of $\mathbf{co}_{xy}$ and $\mathbf{sc}_y$ converges to 0 on each iteration. The PSC update equation can consequently be simplified to incorporate only the self-organizing term. Notice that by keeping α and nullifying both β and γ, the PSC generalizes to the k-means with momentum and stochastic learning rate,

$$v(t+1) = \sum_{y \in \mathbb{C}_{\check{x}(t)}} \omega \check{v}(t) + \check{\alpha} \varphi_{so}(y - \check{x}(t)), \tag{4.45}$$

$$= \sum_{y \in \mathbb{C}_{\check{x}(t)}} \overbrace{\frac{\omega v(t)}{|\mathbb{C}_{\check{x}(t)}|}}^{momentum} + \overbrace{\frac{\alpha \varphi_{so}}{|\mathbb{C}_{\check{x}(t)}|}}^{learning\,rate} (y - \check{x}(t)), \tag{4.46}$$

$$x(t+1) = \sum_{y \in \mathbb{C}_{\check{x}(t)}} \check{x}(t) + \check{v}(t+1), \tag{4.47}$$

$$= \overbrace{x(t)}^{initial\,position} + \overbrace{\frac{1}{|\mathbb{C}_{x(t)}|} \sum_{y \in \mathbb{C}_{x(t)}} \omega v(t) + \alpha \varphi_{so}(y - x(t))}^{resultant\,vector}. \quad ■ \tag{4.48}$$

Case 1: $\alpha = 0$

When $\alpha = 0$, a problematic condition emerges. Without the self-organizing vector and zero initial velocity (e.g. $\alpha = 0$ and $v(0) = 0$), particles will only be attracted to its starting position and assume zero velocity throughout. When $\alpha = 0$ and $v(0) \neq 0$, the trajectory of the particle is

determined by its momentum or residual velocity. This obvious case is seen as follows,

$$\begin{aligned} v(t+1) &= \omega(t)v(t) + \overbrace{\alpha}^{=0}\varphi_{so}\overbrace{(y - x(t))}^{so_y(t)} + \beta\varphi_{co}\overbrace{(p_{xy}(t) - x(t))}^{co_{xy}(t)} + \gamma\varphi_{sc}\overbrace{(g_y(t) - x(t))}^{sc_y(t)} \\ &= \omega(t)v(t) + \beta\varphi_{co}\overbrace{(p_{xy}(t) - x(t))}^{co_{xy}(t)} + \gamma\varphi_{sc}\overbrace{(g_y(t) - x(t))}^{sc_y(t)} \\ v(t=1) &= \overbrace{\omega(0)v(0)}^{\substack{\text{must be } > 0, \\ \text{otherwise } v(t>0) = v(0) = 0}} + \beta\varphi_{co}\overbrace{(x(0) - x(0))}^{co_{xy}(0)=0} + \gamma\varphi_{sc}\overbrace{(x(0) - x(0))}^{so_y(0)=0}, \end{aligned} \tag{4.49}$$

In case of $v(0) \neq 0$ and $\alpha = 0$, each particle is expected to oscillate around $x(0)$ with exponentially decreasing velocity, until it finally stagnates as the residual velocity depletes. The personal and global best will be updated if and only if the angle between the velocity vector $v(t)$ and $p_{xy}(t) - y$ is less than 90 degrees. From this observation we show that the value for β and γ are ineffective if $\alpha = 0$.

The PSC guarantees convergence as long as α is nonzero and inside the convergence bounds as described in Equation 4.37. The resultant vector of the PSC movement generalizes to a vector of maximum likelihood estimate for each $y \in \mathbb{C}_x$ when seen at a higher level. An interesting observation is that zero values in β and γ has negligible effect in term of convergence. Hence Theorem 4.1 suggests:

1. β and γ are redundant, and
2. PSC as sensitive to initialization as k-means.

In 2010, Szabo et al. performed an empirical analysis with regards to the effect of α, β, and γ to the trajectory of PSC particles where they observed this particular convergence phenomenon [85]. Based on the experiments presented in their manuscript, Szabo et al. suggested that *"there were no much gain obtained with the PSC when compared with a standard self-organizing clustering methods"* — i.e. when α is held constant, $\beta = \gamma = 0$. However, the reason behind the phenomenon were not explained. Theorem 4.1 therefore provide the theoretical proof necessary to complete their analysis [85].

4.4.2 Particle Behavior

We generate a two dimensional artificial dataset using the equation of a circle: $y_1(\theta) = r_1\cos(\theta) + c_1$ and $y_2(\theta) = r_2\sin(\theta) + c_2$. Three non-overlapping circles are generated by varying $\mathbf{r}$ and $\mathbf{c}$, representing three clusters. The behavior of each particle on this dataset under various parameter

settings is shown in Figure 4.4. The consequence of Theorem 4.1 can be directly observed in this figure. It can be observed that when α is zero, the values of β and γ have no effect. This is obvious particularly in Figure 4.4c and Figure 4.4d where all particles stay stationary in its initial position. Specifying α, β, and γ from the stability bound in Equation 4.31 results in a cyclical movement as expected. Specifying α, β, and γ based on the stability bound specified in Equation 4.37, each particle finally converges to the center of each cluster.

Another experiment were done using the five Gaussian dataset where the algorithm was executed using parameters specified by the stability bounds in Figure 4.5, with $\alpha > 0$ and a randomized β and γ (inside the stability bounds). After numerous trials the trajectory of the particles on this dataset were as shown in Figure 4.5 which clearly shows the PSC's sensitivity to initialization.

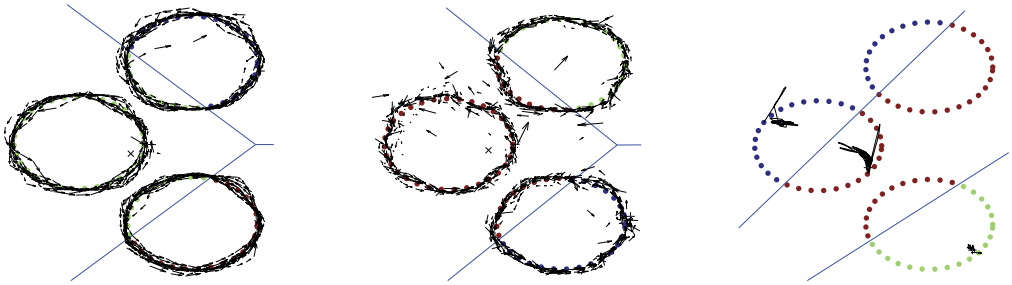
All PSC families, including the original Cohen - de Castro's PSC [75], Szabo's mPSC [74], and Szabo's FPSC [70], share the behavior depicted in Figure 4.4 and Figure 4.5. Based on their this observation, we propose that the three algorithms, although proposed under varying names and time frames, are conceptual duplicates of the original PSC.

4.5 Performance Analysis

The performance of the algorithms of the PSC families were evaluated against Fisher's Iris dataset [163], Wine dataset, and Spam Emails dataset, all of which are openly available from the UCI machine-learning repository [84]. The machine used in this simulation is a laptop computer with Intel Core i5 M520 @ 2.4 Ghz, 4 GB of RAM, running Windows 7. All simulations were done in MATLAB 7.10.0 (R2010a).

The performance were investigated using two internal cluster validation indices: the Generalized Dunn Index $\mathfrak{V}_{GDN}$ and Simplified Silhouette Criterion $\mathfrak{V}_{SSWC}$; and two external cluster validation indices: Purity $\mathfrak{V}_{P}$ and Normalized Mutual Information $\mathfrak{V}_{NMI}$. A detailed explanation on the cluster validation indices are provided in Section 2.4.

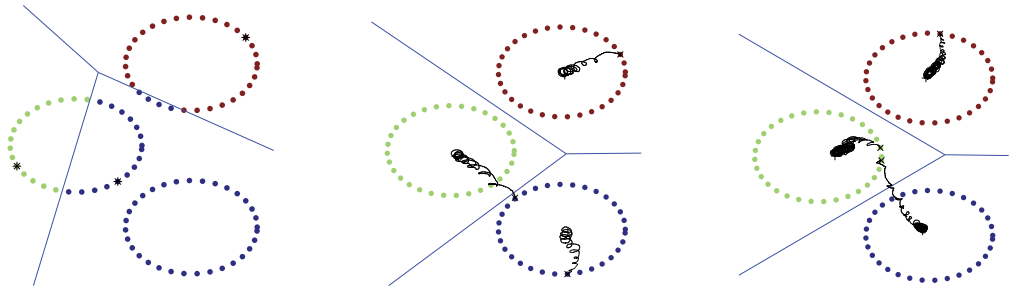
The competing algorithms were the k-means++, Fuzzy c-means, and RCE^{r+} (2014) [55]. All algorithms were fairly selected as they are all based on the voronoi tessellation principles. The parameters for the PSC families were set as follows: $\alpha = \beta = \gamma = K/|\mathbb{Y}|$, $iter_{max} = 200$. Each experiment was repeated 300 times.



(A) $\alpha = 1, \beta = 0, \gamma = 0, \omega = 0.5$, Cyclical trajectory, does not converge.

(B) $\alpha = 1, \beta = 0.5, \gamma = 0.5, \omega = 0.5$, Cyclical trajectory, does not converge.

(C) $\alpha = 0, \beta = 1, \gamma = 1, \omega = 0.5$, stationary, does not converge.

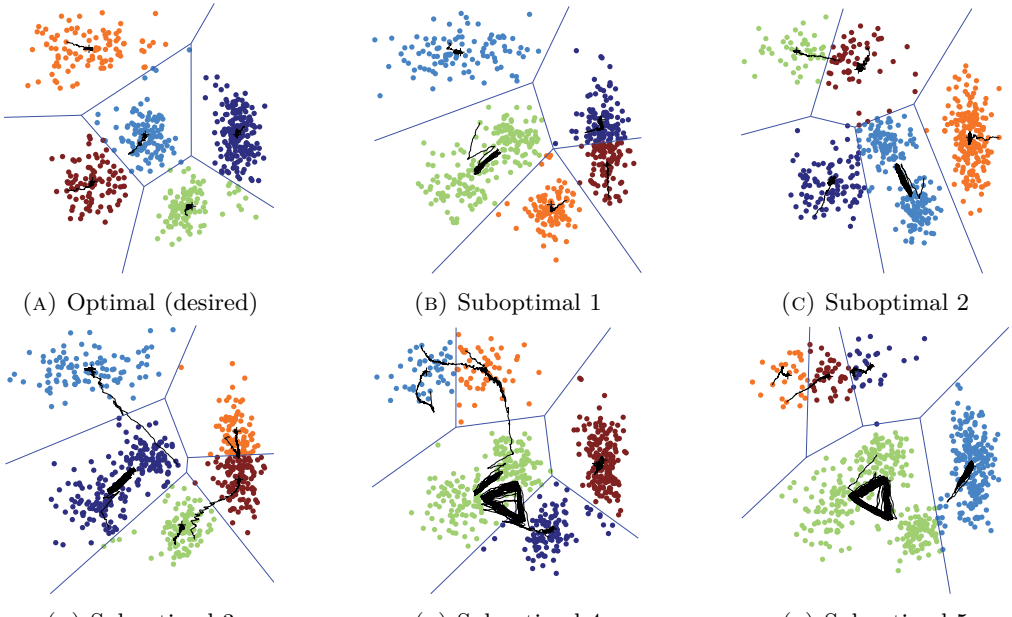


(D) $\alpha = 0, \beta = 0, \gamma = 1, \omega = 0.5$, $v(0) = 0$ oscillates around initial position, does not converge.

(E) $\alpha = 0.01, \beta = \gamma = 0, \omega = 0.5$, $\{\alpha, \beta, \gamma\} < \frac{2+2\omega}{|\mathbb{C}_x|}$, converges according to the maximum likelihood (k-means) trajectory. Sensitive to initial position.

(F) $\alpha = 0.01, \beta = \gamma = 0.005, \omega = 0.5$, $\{\alpha, \beta, \gamma\} < \frac{2+2\omega}{|\mathbb{C}_x|}$, converges according to the maximum likelihood (k-means) trajectory. Sensitive to initial position.

FIGURE 4.4: Trajectory of the PSC/mPSC/FPSC particles on an artificial dataset.



(A) Optimal (desired)

(B) Suboptimal 1

(C) Suboptimal 2

(D) Suboptimal 3

(E) Suboptimal 4

(F) Suboptimal 5

FIGURE 4.5: Trajectory of the PSC/mPSC/FPSC particles on the five Gaussian dataset with numerous random seeding shows PSC's sensitivity to initialization.

4.5.1 Fisher - Iris Dataset

4.5.1.1 General Overview

The Fisher-Iris dataset [163] contains 150 instances of iris flowers collected in Hawaii. The dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured including sepal length, sepal width, petal length, and petal width.

The p-value matrix for testing the hypothesis of no correlation between features/classes can be seen in Table 4.2. The distribution of the features can be seen in the plot matrix in Figure 4.6. It can be seen from these information that even though the features are highly correlated with one another, each feature encodes additional information that would assist with the species identification task. For example, at least one of the petal information (either length or width) needs to be retained to enable reliable segmentation of Iris virginica and Iris versicolor.

TABLE 4.2: Iris Features: P-value Matrix for testing the hypothesis of no correlation against the alternative that there is a non-zero correlation.

	Sepal Length	Sepal Width	Petal Length	Petal Width	Setosa	Virginica	Versicolor
Sepal Length		0.18	<0.01	<0.01	<0.01	0.33	<0.01
Sepal Width			<0.01	<0.01	<0.01	<0.01	0.11
Petal Length				<0.01	<0.01	0.01	<0.01
Petal Width					<0.01	0.15	<0.01

4.5.1.2 Choosing the Distance Quantifier

Botanists characterize the species of iris flowers by looking at the morphological characteristics of their sepals and petals [163]. The intraspecies variability with regards to the size of the petals and sepals are linearly proportionate to the flower's overall size. This variability is captured accordingly in Fisher's Iris dataset [163], where the flowers in each species can be seen probabilistically distributed in an elongated multivariate Gaussian mixture.

In the scatter plot matrix in Figure 4.6, it can be observed that Iris Setosa is linearly separable to the two others in the euclidean space; while the remaining two slightly overlaps. What makes the Iris dataset problematic is the fact that the data are distributed in rotated hyper-ellipsoids. Globular metrics (e.g. Euclidean, Chebyshev, Manhattan, etc.) are therefore unsuitable for this data. As the intracluster correlation is considerably high, the dataset can be clustered easily using Pearson's correlation. However, correlation distances only specifies the relationship between

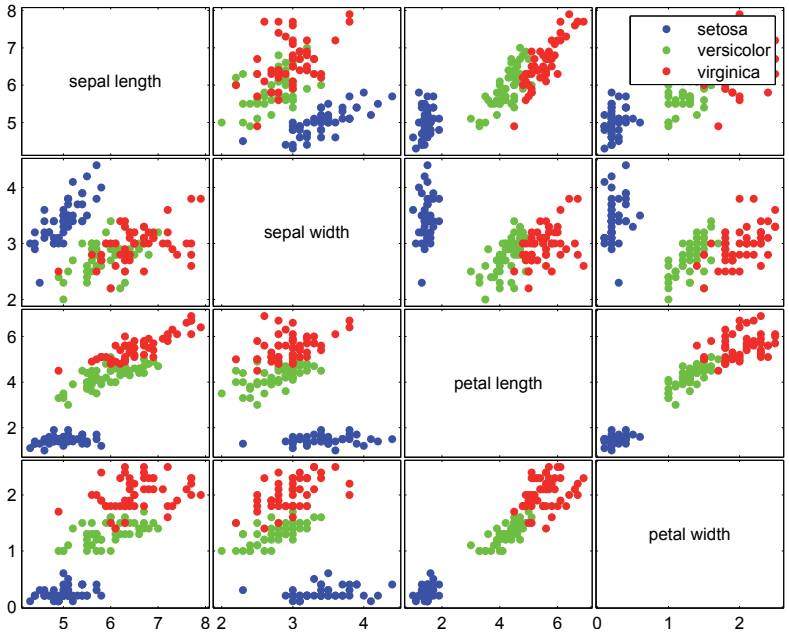


FIGURE 4.6: Fisher-Iris dataset. Measures are in cm.

features but not the extent to which they scale. As it is important to note that the size of a flower has to be normally distributed for each species of Iris, using Pearson's correlation as the clustering metric would lead to a somewhat flawed assumption (e.g. sepals and petals of an iris flower do not grow to ∞ centimeters, neither would their lengths be negative).

Mahalanobis distance allows a clustering algorithm to cluster and consecutively learn the sample within-cluster covariance matrix. As the intraspecies relationship between sepals and petals follows the multivariate normal distribution, the Mahalanobis distance family may be chosen as the appropriate metric for this particular clustering problem. We utilized the Squared Mahalanobis Distance for this experiment due to its convenient quadratic loss property.

4.5.1.3 Cluster Validity Analysis

A successful clustering scenario using the Mahalanobis distance can be seen in Figure 4.7.

The result of various clustering algorithms on this particular dataset using the distance quantifier is shown in Table 4.3. The significance of each algorithm is benchmarked against k-means using Wilcoxon rank sum test for testing equal median. For this dataset, the fuzzification parameter is set to 1.4 for both FCM and FPSC.

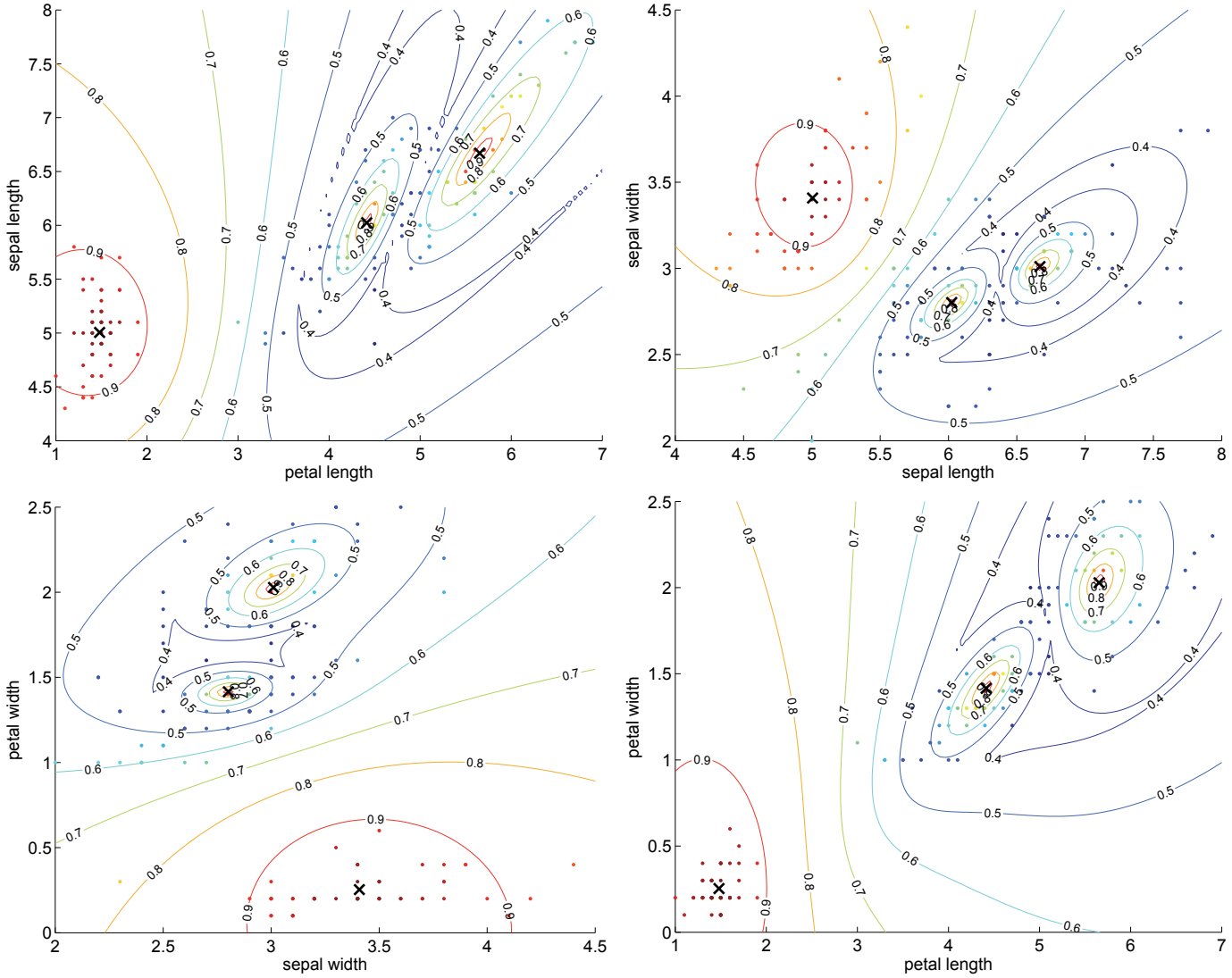


FIGURE 4.7: A Succesful Clustering of the Fisher-Iris dataset ($\mathfrak{V}_P = 0.97$, algorithm: RCE^{r+}). Fuzzification parameter m = 2.

TABLE 4.3: Performance of the PSC families relative to other clustering algorithms on Fisher-Iris dataset (distance quantifier: Squared Mahalanobis Distance). Bold font denotes statistically significant results: black font indicates higher median; red font indicates lower median.

	$\mathfrak{V}_{SSWC}$					$\mathfrak{V}_{GDN}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.46	0.47	0.46	0.53		0.57	0.52	0.36	0.72	
fuzzy c-means	0.48	0.49	0.36	0.63	0.57	0.59	0.57	0.24	0.81	0.84
PSC	0.74	0.75	0.74	0.75	**<0.01**	0.43	0.44	0.42	0.50	**<0.01**
mPSC	0.75	0.75	0.74	0.76	**<0.01**	0.52	0.51	0.46	0.60	0.59
FPSC	0.74	0.75	0.73	0.75	**<0.01**	0.44	0.44	0.42	0.47	**<0.01**
RCE^{r+}	0.78	0.79	0.77	0.80	**<0.01**	1.74	1.81	1.57	1.98	**<0.01**

	$\mathfrak{V}_{Purity}$					$\mathfrak{V}_{NMI}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.85	0.87	0.85	0.89		0.72	0.72	0.69	0.74	
fuzzy c-means	0.81	0.87	0.80	0.89	0.20	0.72	0.74	0.70	0.75	0.45
PSC	0.64	0.62	0.61	0.67	**<0.01**	0.62	0.62	0.61	0.63	**<0.01**
mPSC	0.67	0.64	0.59	0.72	**<0.01**	0.64	0.63	0.62	0.65	**<0.01**
FPSC	0.62	0.62	0.59	0.66	**<0.01**	0.62	0.62	0.61	0.63	**<0.01**
RCE^{r+}	0.95	0.96	0.95	0.96	**<0.01**	0.84	0.85	0.83	0.87	**<0.01**

	Time (ms)		Memory
	μ	σ	(Bytes)
k-means++	5	3.7	12942
fuzzy c-means	6	4.8	12554
PSC	1883	651	38073
mPSC	1739	691	37913
FPSC	1857	767	38097
RCE^{r+}	361	63	14753

The results from Table 4.3 shows that on the Fisher-Iris dataset, only RCE^{r+} achieved higher median in all performance aspects when compared with k-means. PSC, mPSC, and FPSC performed poorly on this dataset, further investigation revealed that the algorithm frequently assigned the two overlapping classes as a single cluster. Although this grouping is somewhat acceptable in clustering point of view (denoted by the higher $\mathfrak{V}_{sswc}$), in terms of the external validity measures this phenomenon were considered as a degradation in performance. The highest median was achieved by RCE^{r+} with 0.96 purity. Further insights on the RCE^{r+} will be covered in the next chapter.

In terms of time and memory complexity, the PSC families were considerably more expensive in comparison to k-means, fuzzy c-means and RCE^{r+}. The PSC families required around 1800ms and 38 Kilobytes of memory on average on this dataset. RCE^{r+} required 361ms and 14.7 Kilobytes of memory. The lowest complexity was achieved by both k-means and fuzzy c-means with 5ms and 13 Kilobytes of memory.

4.5.2 Wine Dataset

4.5.2.1 General Overview

The wine dataset summarizes the results of a chemical analysis of wines grown in the same region in Italy, derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. This dataset is convex, relatively well behaved and linearly separable, which makes it suitable for testing the repeatability of a clustering algorithm. This dataset and its description can be downloaded from [84].

The wine dataset contains 178 wines with 13 features as listed in Table 4.4. Table 4.4 presents the P-value Matrix for testing the hypothesis of no correlation. From this table it is easy to observe that many of the features are redundant. The scatter plot matrix showing only 6 of the 13 features can be seen in Figure 4.8.

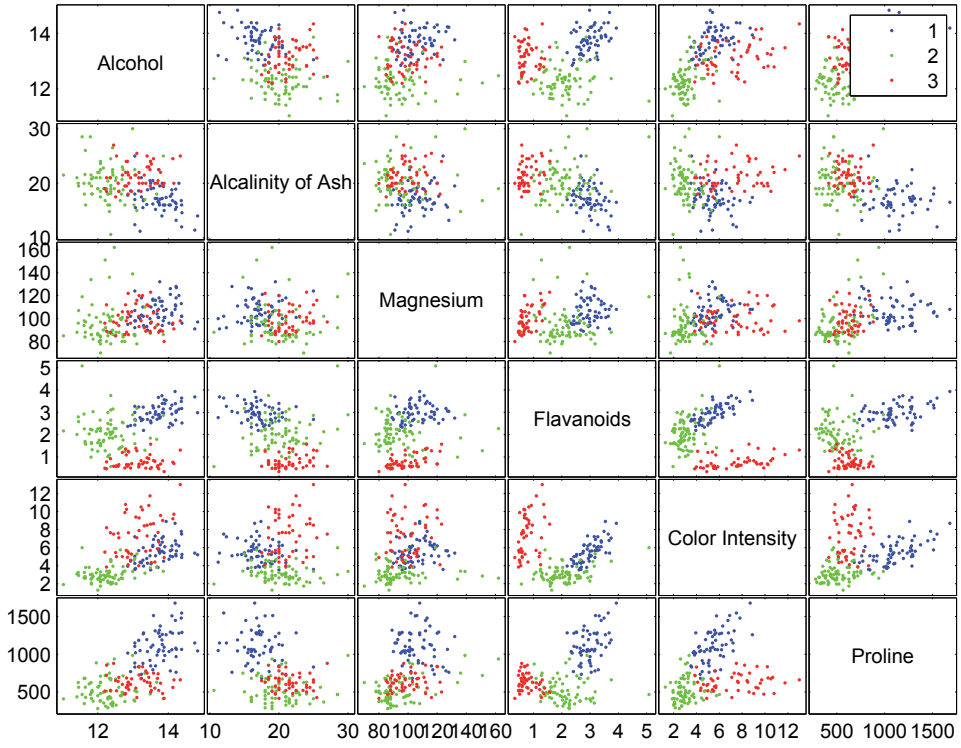


FIGURE 4.8: Six features out of the 13 from the wine dataset.

Figure 4.8 shows that the wines from the first, second, and third cultivars can be clustered using only 6 of the available features. Even though the best practice would be to select only the features that are not redundant, for this experiment we used all 13 features to investigate the robustness of each clustering algorithm when dealing with redundant information.

Table 4.4: Wine Features: P-value Matrix for testing the hypothesis of no correlation against the alternative that there is a non-zero correlation.

		Alcohol	Malic Acid	Ash	Alcalinity of Ash	Magnesium	Total Phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315	Proline	Cultivar A	Cultivar B	Cultivar C
1.	Alcohol		0.21	<0.01	<0.01	<0.01	<0.01	<0.01	0.04	0.07	<0.01	0.34	0.34	<0.01	<0.01	<0.01	0.13
2.	Malic Acid			0.03	<0.01	0.47	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	0.01	<0.01	<0.01	<0.01
3.	Ash				<0.01	<0.01	0.09	0.13	0.01	0.90	<0.01	0.32	0.96	<0.01	<0.01	<0.01	0.04
4.	Alcalinity of Ash					0.27	<0.01	<0.01	<0.01	<0.01	0.80	<0.01	<0.01	<0.01	<0.01	0.02	<0.01
5.	Magnesium						<0.01	<0.01	<0.01	<0.01	<0.01	0.46	0.38	<0.01	<0.01	<0.01	0.81
6.	Total Phenols							<0.01	<0.01	<0.01	0.46	<0.01	<0.01	<0.01	<0.01	0.53	<0.01
7.	Flavanoids								<0.01	<0.01	0.02	<0.01	<0.01	<0.01	<0.01	0.58	<0.01
8.	Nonflavanoid phenols									<0.01	0.06	<0.01	<0.01	<0.01	<0.01	0.88	<0.01
9.	Proanthocyanins										0.74	<0.01	<0.01	<0.01	<0.01	0.46	<0.01
10.	Color intensity											<0.01	<0.01	<0.01	0.06	<0.01	<0.01
11.	Hue												<0.01	<0.01	<0.01	<0.01	<0.01
12.	OD280/OD315													<0.01	<0.01	<0.01	<0.01
13.	Proline														<0.01	<0.01	<0.01

4.5.2.2 Choosing the Distance Quantifier

The scatter plot matrix in Figure 4.8 shows that, due to the correlated features, the wines are distributed in a combination between globular and rotated multivariate Gaussians. This phenomenon can be observed in scatter plot between proline and color intensity; alcohol and flavanoids; flavanoids and proline.

We selected the Jensen-Shannon (JS) distance (square root of the Jensen-Shannon Divergence) as the distance quantifier to cope with these redundancies. The features are first converted into cumulative probability distribution by fitting the beta distribution on each normalized feature.

The cumulative distribution calculates the area under the beta curve which is particularly useful for this dataset because it directly quantifies the cumulative probability $p(x_i < x_{ij}|\theta_\beta)$ — i denotes the feature index and j denotes the observation index. In terms of magnesium content, for example, it can be conveniently translated as the probability of getting a type of wine with lower magnesium concentration than the wine at hand, given that we know the distribution of magnesium over all wines in the dataset.

4.5.2.3 Cluster Validity Analysis

A successful clustering scenario using the JS distance can be seen in Figure 4.9.

The result of various clustering algorithms on this particular dataset using the distance quantifier is shown in Table 4.5. The significance of each algorithm is benchmarked against k-means using Wilcoxon rank sum test for testing equal median. For this dataset, the fuzzification parameter is set to 1.4 for both FCM and FPSC.

The results from Table 4.5 shows that on the Wine dataset, only RCE^{r+} achieved a statistically significant higher median on both internal cluster validity indices ($\mathfrak{V}_{SSWC}$ and $\mathfrak{V}_{GDN}$) compared to k-means. All other algorithms, including Fuzzy c-means, PSC, mPSC, and FPSC, achieved lower $\mathfrak{V}_{SSWC}$ on this dataset. With regards to $\mathfrak{V}_P$ Although there is no significant difference in terms of median between RCE^{r+} and k-means, the purity average ($\mu(\mathfrak{V}_P)$) of RCE^{r+} is closer to the median compared to that of k-means, which implies that k-means has a slightly skewed distribution due to its heavier left tail. Nevertheless, all algorithms achieved a satisfactory median purity up to as high as 0.97 on this dataset through a completely unsupervised process. The result shows that the appropriateness of use of the cumulative probability and β-distribution fitting scheme together with the Jensen-Shannon distance.

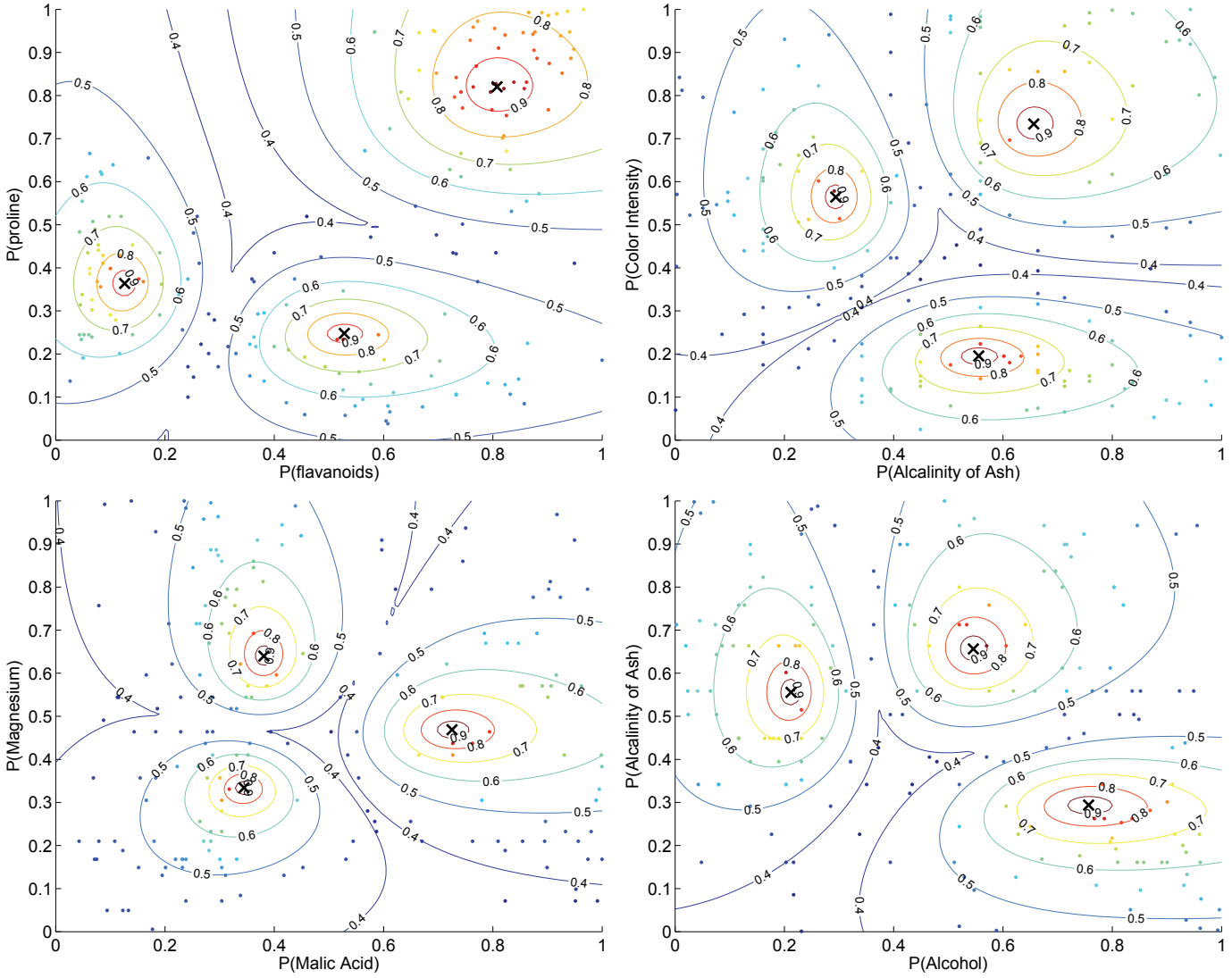


FIGURE 4.9: A Succesful Clustering of the Wine dataset using Jensen-Shannon distance ($\mathfrak{V}_P = 0.97$, algorithm: $\text{RCE}^{\text{r}+}$). Values are expressed in terms of cumulative probability. Fuzzification parameter m = 2.

TABLE 4.5: Performance of the PSC families relative to other clustering algorithms on Wine dataset (distance quantifier: Jensen-Shannon distance). Bold font denotes statistically significant results: black font indicates higher median; red font indicates lower median.

	$\mathfrak{V}_{SSWC}$					$\mathfrak{V}_{GDN}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.454	0.457	0.457	0.46		0.705	0.716	0.716	0.72	
fuzzy c-means	0.339	0.363	0.363	0.36	**<0.01**	0.579	0.607	0.607	0.61	**<0.01**
PSC	0.446	0.448	0.440	0.46	**<0.01**	0.709	0.713	0.697	0.74	0.24
mPSC	0.425	0.439	0.429	0.46	**<0.01**	0.672	0.709	0.681	0.74	0.06
FPSC	0.442	0.446	0.437	0.45	**<0.01**	0.694	0.709	0.690	0.72	0.03
RCE^{r+}	0.459	0.458	0.457	0.46	**<0.01**	0.722	0.719	0.717	0.72	**<0.01**

	$\mathfrak{V}_{Purity}$					$\mathfrak{V}_{NMI}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.955	0.972	0.972	0.972		0.882	0.897	0.897	0.897	
fuzzy c-means	0.949	0.972	0.972	0.972	0.02	0.866	0.897	0.897	0.897	0.02
PSC	0.958	0.966	0.961	0.972	**<0.01**	0.870	0.878	0.859	0.897	**<0.01**
mPSC	0.908	0.961	0.949	0.966	**<0.01**	0.822	0.862	0.836	0.880	**<0.01**
FPSC	0.956	0.972	0.966	0.972	**<0.01**	0.872	0.893	0.878	0.897	**<0.01**
RCE^{r+}	0.971	0.972	0.972	0.972	0.73	0.896	0.897	0.897	0.897	0.73

	Time (ms)		Memory
	μ	σ	(Bytes)
k-means++	15.68	6.00	19559
fuzzy c-means	147.14	85.80	24649
PSC	5733.61	1360.47	102619
mPSC	5174.45	1356.52	103031
FPSC	3767.97	1380.34	100986
RCE^{r+}	556.88	103.91	33832

In terms of time and memory complexity, the PSC families were considerably more expensive in comparison to k-means, fuzzy c-means and RCE^{r+}. The PSC families required up to 5700 ms and 103 Kilobytes of memory on average on this dataset. RCE^{r+} required 556.88 ms and 33.8 Kilobytes of memory. The lowest complexity was achieved by both k-means and fuzzy c-means with 15ms and 19.5 Kilobytes of memory.

4.5.3 Spam E-mail Dataset

4.5.3.1 General Overview

The spam email dataset is originally donored by George Forman from Hewlett-Packard for the development of a reliable spam filtering. The dataset is available online from the UCI machine learning dataset [84]. The dataset contains 57 features which are extracted from 4601 emails (2788 spam emails, and 1813 non-spam emails). The features can be summarized in Table 4.6. The task

in this dataset is straightforward, which is to classify spam and non-spam emails based on the available features.

TABLE 4.6: Spam dataset feature overview, as reported in the original dataset

Feature type	Description
48 continuous real [0,100] attributes of type word_freq_WORD	percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
6 continuous real [0,100] attributes of type char_freq_CHAR	percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail
1 continuous real [1,...] attribute of type capital_run_length_average	average length of uninterrupted sequences of capital letters
1 continuous integer [1,...] attribute of type capital_run_length_longest	length of longest uninterrupted sequence of capital letters
1 continuous integer [1,...] attribute of type capital_run_length_total	sum of length of uninterrupted sequences of capital letters

It is important to understand that this dataset is normally used for supervised learning. In this experiment we are interested on whether this classification problem can be solved using an unsupervised process. The performance will therefore depend on the landscape or distribution of the data, the appropriateness of the distance measure and the efficiency of the clustering algorithm.

This dataset is relatively bigger than iris and wine dataset, with an approximate size of 2 megabytes. From the performance of PSC families on the past datasets, it is to be expected that the PSC families would perform very slowly on this dataset due to their high complexities.

4.5.3.2 Choosing the Distance Quantifier

In a situation where one is required to filter meaningful emails from spams, it is natural to see "frequency of a word" as a probabilistic measure instead of a number because one does not simply count how many spam keywords are there in an email and compare their repetitions against any predefined threshold. The natural way is rather to investigate whether certain "keywords" appear or even dominate the content of the email. The emails whose observable patterns share similar characteristics to those marked as spam are then more likely to be assigned in the spam group. Based on this analysis we believe that the Symmetrical Kullback Leibler (KL) divergence with the column-normalized feature (e.g. each column sums to 1) may be an appropriate distance quantifier for this dataset. The column-normalization is essential to convert the feature vector into a probability mass function (PMF) where $p_{\mathrm{X}}(x) : \mathrm{X} \to [0, 1, 2, \ldots, 57]$.

Similarly to the wine-dataset, the features in the spam dataset needs to be converted into cumulative distribution function (CDF) by fitting the beta distribution on each feature. However, the data needs to be preprocessed prior to the conversion. The details are as follows.

The features in this dataset are excessively leptokurtic, which is to be expected because contextually the probability of "absence of a word" has to be higher than "repetition of a word" due to the following. Consider two specific words, say "meeting" and "3d", where "meeting" has been used in an example email conversation. Let us further assume that the word "3d" is out of context. On one hand, the word "meeting" will naturally repeat during the course of the conversation to preserve contextual coherence. On the other hand, the word "3d" is nonexistent, simply because it is out of the context of the conversation. Based on this analysis we apply logarithmic transformation (e.g. $\log(1+x)$) to each feature such that higher frequency repetitions are assigned *fairer* weights compared to absence. These log-transformed features are then scaled from 0 to 1 and fitted with the beta distribution.

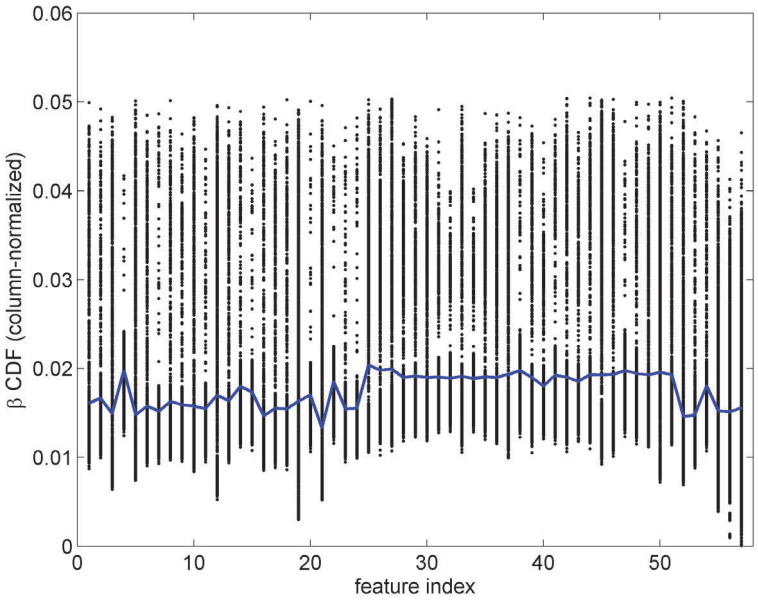
4.5.3.3 Cluster Validity Analysis

A successful clustering scenario using the Symmetrical KL-divergence can be seen in Figure 4.10.

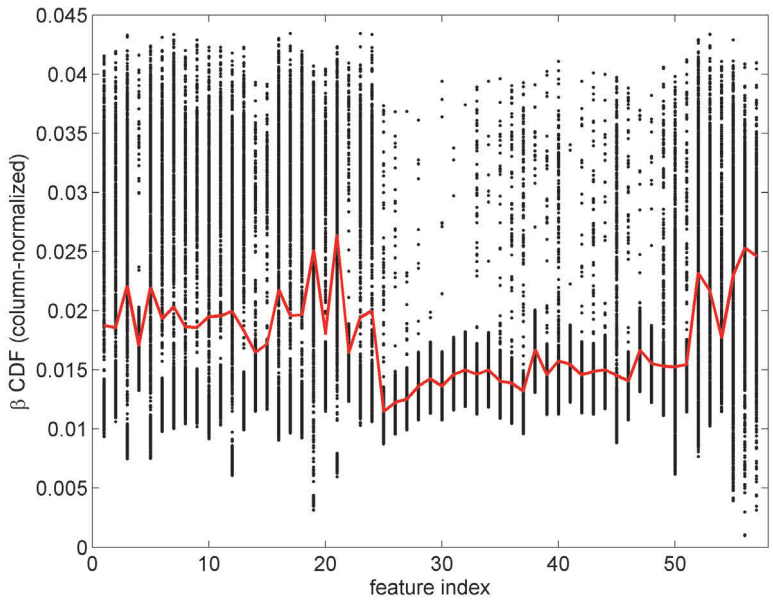
The result of various clustering algorithms on this particular dataset using the distance quantifier is shown in Table 4.5. The significance of each algorithm is benchmarked against k-means using Wilcoxon rank sum test for testing equal median. For this dataset, the fuzzification parameter is set to 1.4 for both FCM and FPSC.

The results from Table 4.5 shows that on the this dataset, only $\text{RCE}^{\text{r}+}$ achieved higher median in all cluster validity indices compared to k-means. Nevertheless, all algorithms achieved a satisfactory purity up to as high as 0.88 on this dataset through a completely unsupervised process which shows that the task can be elegantly tackled with the appropriate use of cluster analysis.

In terms of time and memory complexity, $\text{RCE}^{\text{r}+}$ were slightly more expensive than fuzzy c-means (8 seconds, 2.3 MB memory compared to 6.6 seconds 2.2 MB memory). The lowest complexity was achieved by k-means with 381ms and 2.1 Megabytes of memory. The PSC families were considerably more expensive, inefficient, yet yield poorer results in comparison to the other algorithms on this dataset. The PSC families required up to 200 seconds at the worst case with as high as 8.5 Megabytes of memory (about 4.25 times the size of the dataset itself).



(A) Clustered non-spam emails: Individual samples and the learned Probability Mass Function (PMF).



(B) Clustered spam emails: Individual samples and the learned PMF.

FIGURE 4.10: A Succesful Clustering of the Spam dataset using Symmetrical Kulback-Leibler Divergence ($\mathfrak{V}_P = 0.901$, algorithm: $\text{RCE}^{\text{r}+}$), thick line in each graph denotes the learned PMF, which is the probabilistic barycenter of the 57-dimensional KL Voronoi region inferred from the data using $\text{RCE}^{\text{r}+}$. Values are expressed in terms of column-normalized β-CDF.

TABLE 4.7: Performance of the benchmarked clustering algorithms on Spam E-mail dataset (distance quantifier: Symmetrical Kullback-Leibler Divergence). Bold font denotes statistically significant results: black font indicates higher median; red font indicates lower median.

	$\mathfrak{V}_{SSWC}$					$\mathfrak{V}_{GDN}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.417	0.396	0.394	0.40		0.584	0.609	0.559	0.61	
fuzzy c-means	0.229	0.226	0.226	0.23	**<0.01**	0.516	0.514	0.514	0.51	**<0.01**
PSC	0.329	0.315	0.290	0.36	**<0.01**	0.614	0.615	0.586	0.65	**<0.01**
mPSC	0.332	0.329	0.287	0.37	**<0.01**	0.607	0.607	0.586	0.62	0.10
FPSC	0.297	0.293	0.225	0.34	**<0.01**	0.578	0.575	0.558	0.60	**<0.01**
RCE^{r+}	0.400	0.400	0.399	0.40	**<0.01**	0.662	0.662	0.661	0.66	**<0.01**

	$\mathfrak{V}_{Purity}$					$\mathfrak{V}_{NMI}$				
	μ	median	Q_1	Q_3	p	μ	median	Q_1	Q_3	p
k-means++	0.783	0.885	0.524	0.887		0.389	0.484	0.167	0.488	
fuzzy c-means	0.858	0.858	0.858	0.858	**<0.01**	0.401	0.399	0.399	0.399	**<0.01**
PSC	0.836	0.845	0.814	0.881	**<0.01**	0.386	0.385	0.338	0.466	**<0.01**
mPSC	0.833	0.862	0.796	0.879	**<0.01**	0.392	0.418	0.311	0.471	**<0.01**
FPSC	0.857	0.866	0.830	0.882	**<0.01**	0.421	0.444	0.365	0.468	**<0.01**
RCE^{r+}	0.900	0.900	0.900	0.901	**<0.01**	0.520	0.522	0.520	0.523	**<0.01**

	Time (ms)		Memory
	μ	σ	(Bytes)
k-means++	780.11	381.31	2109149
fuzzy c-means	6586.65	3818.63	2219739
PSC	160528.87	1299.87	8543234
mPSC	155403.11	2103.55	8541410
FPSC	176858.25	27389.62	8543250
RCE^{r+}	8096.43	216.52	2302100

4.6 Conclusion

This chapter focuses on the analysis of Particle Swarm Clustering variants including PSC, modified PSC (mPSC), and Fuzzy PSC (FPSC). Thorough theoretical investigation and empirical validation has been conducted. Our contributions are summarized as follows.

We have analyzed the theoretical and empirical computational and memory complexity of PSC in Section 4.3 where we unravel a number of important efficiency issues which restricts the algorithm applicability to only smaller datasets. For example, with a 400x1000 data matrix, on each iteration the PSC already suffers from a significant 2000% slow-down compared to k-means. In order to cluster 1000 observations of 1000 dimensional double precision data into 256 clusters, the algorithm variants require as much as 2GB memory; whereas k-means, fuzzy c-means, and Rapid Centroid Estimation (RCE, [55]) require memory allocation of lower than 15MB. Ironically, empirical results on benchmark data from [84] presented in Table 4.3, Table 4.5, and Table 4.7 show that these high complexities do not generally translate into any increase in cluster quality.

We derived the stability criteria and provided a detailed proof of convergence for the PSC algorithm in Theorem 4.1. The proof reveals a problematic cyclical trajectory of the particles using parameter selection outlined in the standard PSO (Lemma 3.1.1). Interestingly this crucial character is discussed neither in the initial manuscript [75] nor its follow ups [70, 74, 85].

When the convergence bounds described in Equation 4.37 are properly satisfied, an even more interesting phenomenon is observed. PSC guarantees convergence as long as the self-organizing constant is nonzero. Furthermore, both social and cognitive parameters have negligible effect to the swarm's convergence, which implies that these parameters are redundant. The ultimate implication of this theorem is that the PSC generalizes to the k-means, retaining all of its performance aspects including the most severe: the curse of initial position. Szabo et al. [85] observed this particular convergence phenomenon in their empirical experiment [85] and suggested that *"there were no much gain obtained with the PSC when compared with a standard self-organizing clustering methods"* [85]. However, the reason behind the phenomenon were not further explained. Theorem 4.1 provides the theoretical proof necessary to complete their analysis.

Finally, empirical testing suggests that all PSC families, including the original PSC [75], Szabo's mPSC [74], and Szabo's FPSC [70], share the exact particle behavior. The three algorithms, although proposed under varying names and time frames, are conceptual duplicates of the original PSC.

Chapter 5

Rapid Centroid Estimation

Rapid Centroid Estimation (RCE) is a semi-stochastic clustering algorithm that we proposed to address the complexity bottleneck of Cohen - de Castro's Particle Swarm Clustering (PSC). The RCE was originally proposed as a lightweight simplification of the PSC algorithm [4, 71–73]. RCE retains the quality of PSC with greatly reduced computational complexity and increased stability. This chapter explains the conceptual journey of the RCE since its prior proposal in 2012 [71, 73], the formulation of the paradigms to address its sensitivity to initialization in 2013 [4, 72], and its further simplification in the 2014 proposition [55].

This chapter is organized as follows. Section 5.1 provides a brief recap on the challenges of the PSC discussed previously in Chapter 2. Section 5.2 defines the basic building blocks. Section 5.3 provides the algorithmic fundamentals. Section 5.4 analyzes the complexity of the RCE algorithm both theoretically and empirically. Section 5.5 analyzes the trajectory and behavior of the RCE particle. Section 5.6 proposes some strategies applicable to reduce the likelihood of the RCE particles converging to suboptimum solution. Section 5.7 summarizes the comparative results of the algorithm on some of the datasets available in UCI machine learning dataset repository [84] as reported in our 2012 publication [4]. Finally, Section 5.8 concludes the chapter.

5.1 The PSC and its Challenges

Despite its excruciatingly expensive computational and memory cost, the PSC suffers the exact same limitations of a standard k-means. As has been analyzed in the previous section, these challenges mainly arise due to the inefficient algorithmic construct and lack of theoretical analysis

on the algorithm. This section reviews the major findings and analysis that we have done in the previous chapter which realization leads to the formulation of the RCE [71, 73].

5.1.1 Computational Complexity

The PSC computational complexity arise mainly due to the external loop for each data. The following comparative summary between a simplified pseudocode of PSC (Algorithm 5.18) vs a simplified pseudocode for k-means (Algorithm 5.19) should explain the reason behind the PSC computational complexity.

Algorithm 5.18 Cohen - de Castro's Particle Swarm Clustering (PSC) - simplified

Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K, PSC swarm parameters.
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

```
1: Initialize the swarm
2: repeat
3:    for all y ∈ 𝕐
4:       Calculate the pairwise distance between 𝕏 and y.
5:       Update the Social and Cognitive matrices,
6:       Generate random vectors φ_so, φ_sc, and φ_co,
7:       update particle velocity and position.
8:    end for
9:    Redirect particles with no member towards the winning particle
10: until Convergence or maximum iteration reached
11: return 𝕏 = {x_1, ..., x_K} ∈ ℝ^dim
```

Algorithm 5.19 K-means - simplified

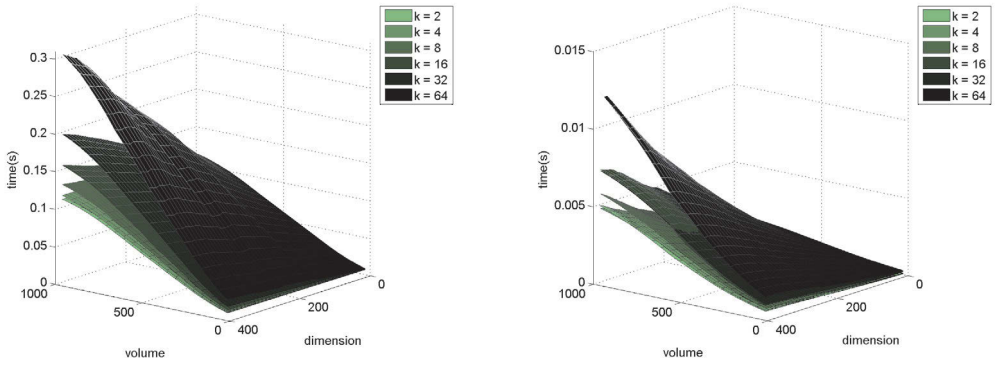
Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

```
1: Initialize position
2: repeat
3:    Calculate the pairwise distance matrix between 𝕏 and 𝕐.
4:    Calculate the crisp membership matrix U
5:    Update 𝕏.
6: until Convergence
7: return 𝕏 = {x_1, ..., x_K} ∈ ℝ^dim
```

It can be easily observed that the bottleneck in the PSC algorithm compared to k-means resides at line 3: **forall** $\mathbf{y} \in \mathbb{Y}$. From the empirical experiments in the previous chapter, we notice that the complexity of line 3 is significant as shown in Figure 5.1. With a 400x1000 data matrix, on each iteration the PSC already suffers from a significant 2000% slow-down compared to k-means.

This time complexity bottleneck inevitably translates to the slow convergence of PSC families in general clustering problems. As the iteration continues the cumulative inefficiency of PSC significantly stretches the overall time required for the algorithm to converge. The experiment

regarding this phenomenon has been done in the previous chapter, which results are summarized as follows.



(A) Time complexity of PSC distance computation.

(B) Time complexity of k-means distance computation.

FIGURE 5.1: Time complexity of the distance computation of PSC vs k-means: Original Figure can be seen in Figure 4.1.

TABLE 5.1: Comparison between time complexities (milliseconds) and median purities of various algorithms on the experiments in the previous chapter. The high time complexities of the PSC families do not generally translate into higher quality result.

	Iris		**Wine**		**Spam Email**	
	Time (ms)	Med $\mathfrak{V}_P$	Time (ms)	Med $\mathfrak{V}_P$	Time (ms)	Med $\mathfrak{V}_P$
k-means++	5	0.873	16	0.972	780	0.885
fuzzy c-means	5	0.873	147	0.972	6587	0.858
PSC	1883	0.623	5734	0.966	160529	0.845
mPSC	1739	0.643	5174	0.961	155403	0.862
FPSC	1857	0.617	3768	0.972	176858	0.866
RCE^{r+}	361	0.957	557	0.972	8096	0.886

5.1.2 Memory Complexity

The algorithms of the PSC families are considerably expensive in terms of memory complexity. Our benchmark test revealed that in order to cluster 1000 observations of 1000 dimensional double precision data, the PSC families require as much as 2GB memory; whereas k-means, fuzzy c-means, and RCE^{r+} require memory allocation of lower than 15MB. More information can be seen in the experiment in the previous chapter, in particular Figure 4.3 and Table 4.1.

This relatively high memory requirement poses a significant scalability bottleneck when dealing with larger, higher dimensional datasets. We summarize the memory requirement of the PSC

families vs k-means fuzzy c-means and RCE^{r+} on the Iris, Wine, and Spam email datasets obtained in the previous chapter.

TABLE 5.2: Comparison between memory complexities (bytes) and median purities of various algorithms on the experiments in the previous chapter. Similarly to the high time complexity, the high memory complexities of the PSC families do not generally translate into higher quality result.

	Iris		**Wine**		**Spam Email**	
	Memory	Med $\mathfrak{V}_P$	Memory	Med $\mathfrak{V}_P$	Memory	Med $\mathfrak{V}_P$
k-means++	12942	0.873	19559	0.972	2109149	0.885
fuzzy c-means	12554	0.873	24650	0.972	2219739	0.858
PSC	38073	0.623	102619	0.966	8543234	0.845
mPSC	37913	0.643	103031	0.961	8541410	0.862
FPSC	38097	0.617	100986	0.972	8543250	0.866
RCE^{r+}	14753	0.957	33832	0.972	2302100	0.886

Particularly in the spam email dataset, the PSC families requires 8.5 Megabytes memory compared to k-means which requires 2.1 Megabytes of memory. The algorithm also performs significantly slower than the other three. We present a complexity map of the benchmarked algorithms on the benchmarked datasets in Figure 5.2. Among the PSC variants, FPSC shows to be the most inefficient. In this Figure we can observe that, despite the complexity, the expensive costs of all the PSC algorithm variants do not generally translates into performance.

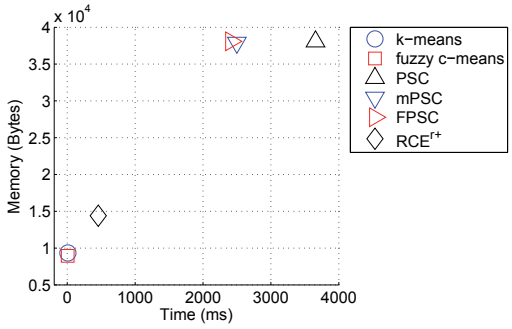
5.1.3 Redundancies and Sensitivity to Initialization

In Theorem 4.1 we have proven the convergence of PSC which validates Cohen - de Castro's proposition on the self-organizing property of the PSC particles [75]. The proof, however, comes with rather intriguing implications.

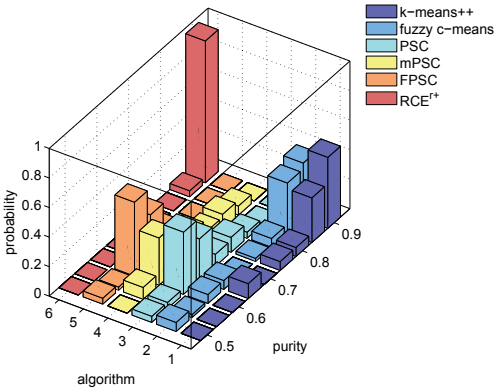
The PSC monotonically minimizes the average distortion between each particle and data in its voronoi region — similarly to Stuart Lloyd's k-means — by following the path of maximum likelihood which resembles the Expectation Maximization (EM) algorithm. The EM is very sensitive to initialization, however it guarantees convergence to a local minimum.

Theorem 4.1 proposes that the PSC generalizes to the EM algorithm (more specifically, the k-means) when the stability constraint in Equation 4.37 is satisfied. Consequently, PSC suffers from the limitations of k-means, including the most severe: initialization and suboptimal partitions.

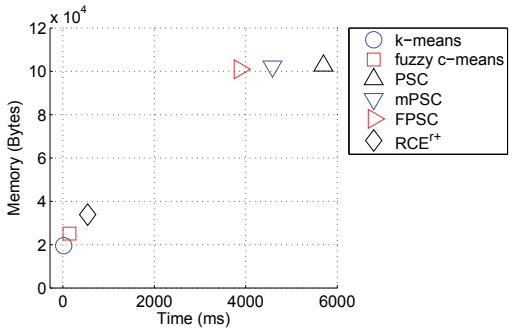
The PSC strongly resembles stochastic k-means which learning rate depends solely on the self-organizing term. Furthermore, both social and cognitive terms can be omitted as their values are



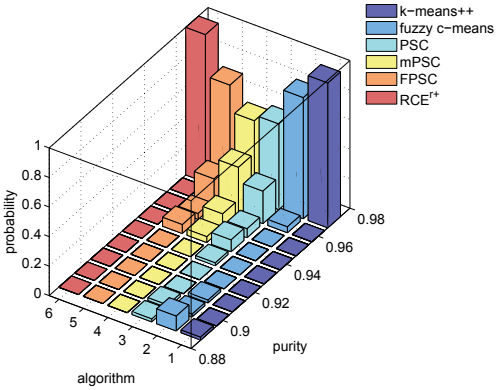
(A) Iris dataset (Squared Mahalanobis distance): complexity map.



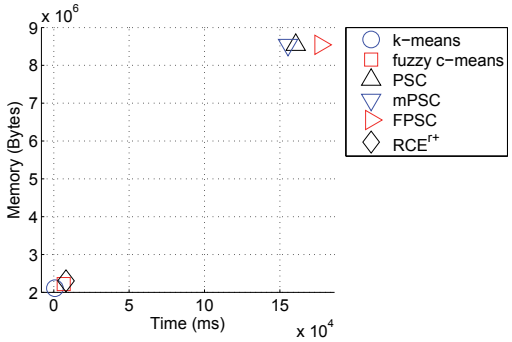
(B) Iris dataset (Squared Mahalanobis distance): purity histogram.



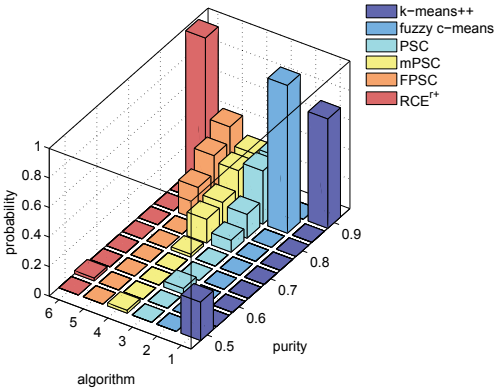
(C) Wine dataset (Jensen-Shannon distance): complexity map.



(D) Wine dataset (Jensen-Shannon distance): purity histogram.



(E) Spam email dataset (Symmetrical Kullback-Leibler divergence): complexity map.



(F) Spam email dataset (Symmetrical Kullback-Leibler divergence): purity histogram.

FIGURE 5.2: Complexity and Purity of the Benchmarked Algorithms.

only effective if and only if the self-organizing term is non-zero. Hence contrast to the claim by Cohen and de Castro [75], we propose that the convergence, and therefore, the performance of PSC is affected only by the self-organizing term, but neither the social nor the cognitive terms. This theoretical proof is in line with the empirical evidence observed by Szabo et al. [85], where they stated that *"there were no much gain obtained with the PSC when compared with a standard self-organizing clustering methods"* [85]. We encourage readers to refer to Theorem 4.1 for an in depth discussion with regards to this matter.

5.2 Definitions and Redefinitions

As can be seen previously in Algorithm 5.18, the PSC computational complexity arise mainly due to the external loop for each data. In RCE 2012 [71, 73], we propose that the PSC algorithm can be further simplified and even improved while still maintaining its algorithmic integrity. In this new construct, the external loop is made compact as a batch matrix operations as in k-means. The movement is updated by summing the *resultant vector* of each term. Although the memory complexity remains the same, the time complexity of each iteration can be significantly reduced [71]. In 2014, the discovery of Theorem 4.1 has unlocked the real potential of RCE, allowing it to process megabyte datasets and engage in consensus swarm clustering setting in quasilinear complexity [55].

In order to properly define the RCE, we first need to equip the abstract construct with the appropriate definitions. One might notice that some of the definitions are redefined from the ones used in PSC, and that the difference may seem very subtle. We would like to emphasize that these subtle differences are important for the proper interpretation of our proposed algorithm. The summary of the (re)definitions includes,

1. A new conceptual interpretation of "swarm" and "particles", and how they interact with one another;
2. A new "modular" perspective on the swarm memory;
3. A computationally efficient movement update scheme which utilizes the resultant vector for each particles instead of individual vectors for each data;
4. A proposition for synthesizing cluster validity index to the PSC as an objective function for quantifying the quality of a voronoi tessellation, and;
5. A local minimum memory which stores the positions of particles that optimizes the given objective function.

The new definitions are explained in detail as follows.

Definition 5.2.1 *(Swarm).* A swarm Θ represents a candidate partition of a dataset $\mathbb{Y} \in \mathbb{R}^{dim}$. The swarm consists of particle tuples $\mathfrak{p} = \{\theta_1, \ldots, \theta_K\}$ and term memory matrices $\mathbb{T}_l = \{\psi_{1\,(l)}, \ldots, \psi_{n(l)}\} \in \mathbb{R}^{dim}$ as follows,

$$\Theta = \{\{\theta_1, \ldots, \theta_K\}, \{\psi_{1\,(1)}, \ldots, \psi_{n\,(1)}\}, \ldots, \{\psi_{1\,(1)}, \ldots, \psi_{n\,(L)}\}\} \tag{5.1}$$

$$= \{\mathfrak{p}, \mathbb{T}_1, \ldots, \mathbb{T}_L\} \tag{5.2}$$

$n(l)$ denotes the number of vectors in the memory matrix. The cardinality of each term memory matrix is determined by the number of vectors it stores. For example, the social memory matrix stores $|\mathbb{T}_{sc}| = |\mathbb{Y}_\Theta|$ vectors; the cognitive memory matrix stores $|\mathbb{T}_{co}| = K \times |\mathbb{Y}_\Theta|$ vectors; the self-organizing memory matrix stores the data vectors $|\mathbb{T}_{so}| = |\mathbb{Y}_\Theta|$; while the local minimum matrix stores $|\mathbb{T}_{mi}| = K$ particle position vectors.

The number of particles, $K = |\mathbb{C}|$, specifies the number of desired voronoi regions $\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_K\}$.

Note that with this new "modular" definition of the swarm, we can easily add or remove memories as if they were modules by specifying the set of terms $\mathbb{L}$. For example, a conservative configuration would be to use all four terms [4, 71, 73]:

$$\mathbb{L}_{PSC,RCE2012} = \{\text{self organizing, social, cognitive, minimum}\} \tag{5.3}$$

If one wish to incorporate the proof from Theorem 4.1, we can obtain an approach as follows,

$$\mathbb{L}_{RCE2014} = \{\text{self organizing, minimum}\}, \tag{5.4}$$

which has been utilized to our proposal of the Ensemble RCE (ERCE) in [55].

Definition 5.2.2 *(Particle).* The particle matrix $\mathfrak{p}$ is a 2-tuple matrix storing the position and velocity vector of each particle tuple $\theta_i = \{\mathbf{x}_i, \mathbf{v}_i\}; (\mathbf{x}, \mathbf{v}) \in \mathbb{R}^{dim}$ such that,

$$\mathfrak{p} = \{\{\mathbf{x}_1, \mathbf{v}_1\}, \ldots, \{\mathbf{x}_K, \mathbf{v}_K\}\}, \tag{5.5}$$

$$= \{\theta_1, \ldots, \theta_K\}, \tag{5.6}$$

$$= \{\mathbb{X}, \mathbb{V}\}, \tag{5.7}$$

where each particle θ_i governs a voronoi region $\mathbb{C}_i$, with voronoi cell $\mathbf{x}_i$. Each data in $\mathbb{C}_i$ is crisply associated with the closest corresponding cell in the Euclidean space defined by the distance function.

Notice that there is a contrast to the original scheme where the cognitive memory is replaced with an external matrix indexed by the swarm. Representing the particles as a 2-tuple allows additional flexibility in adding / removing memories or even parallelizing the swarm [4, 55].

Definition 5.2.3 *(Position).* The position of a particle $\mathbf{x} \in \mathbb{R}^{dim}$ is similarly defined as in PSC where $\mathbf{x}$ denotes its literal location in the *dim*-dimensional Euclidean space. $\mathbf{x}$ represents a potential prototype vector of the voronoi cell. The position of each particle is updated similarly to the standard PSC rule as follows,

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1), \tag{5.8}$$

where $\mathbf{v}$ denotes the velocity vector of the corresponding particle.

Definition 5.2.4 *(Self-Organizing Memory).* The self organizing memory of a swarm is simply a set of observations which are included for the cluster optimization,

$$\mathbb{T}_{so} = \mathbb{Y}_{\Theta}. \tag{5.9}$$

This definition allows the swarm to operate on subsampled/perturbed data. This property is important, especially when the RCE is deployed as parallel cooperative swarm or consensus/ensemble swarm [55].

Definition 5.2.5 *(Cognitive Memory).* Each particle $\mathfrak{p}_i$ is assigned by the swarm a $dim \times N$ cognitive memory $\mathbb{P}_i = \{\mathbf{p}_{i\,1}, \ldots, \mathbf{p}_{i\,|\mathbb{Y}_{\Theta}|}\} \in \mathbb{R}^{dim}$, where each vector in $\mathbb{P}_i$ denotes the closest position of the i^{th} particle in relation to the j^{th} data vector in the self organizing memory $\mathbb{Y}_{\Theta}$. Notice that as each particle is assigned such matrix, the cognitive memory matrix of the swarm $\mathbb{T}_{co}$ can therefore be defined as

$$\mathbb{T}_{co} = \{\mathbb{P}_1, \ldots, \mathbb{P}_K\}, \tag{5.10}$$

where $\mathbb{P}_i$ a $dim \times N$ matrix storing the cognitive memory of the i^{th} particle as accordingly defined. The cardinality of the cognitive memory is therefore $|\mathbb{T}_{co}| = K \times N$.

The cognitive memory is updated as follows,

$$\mathbf{p}_j = \begin{cases} \mathbf{x} & if \quad d(\mathbf{x}, \mathbf{y}_j) < d(\mathbf{p}_j, \mathbf{y}_j) \\ \mathbf{p}_j & otherwise \end{cases}, \tag{5.11}$$

where j denotes the index of data vectors, $d(\cdot, \cdot)$ denotes the distance between two vectors according to a pre-specified distance function.

Definition 5.2.6 *(Social Memory).* The swarm Θ stores the social memory $\mathbb{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_{|\mathbb{Y}_{\Theta}|}\} \in \mathbb{R}^{dim}$ where each vector in $\mathbb{G}$ denotes the position of the particle that has been closest to the

corresponding data vector in the self organizing memory $\mathbb{Y}_{\Theta}$. The social memory matrix of the swarm $\mathbb{T}_{sc}$ is defined as

$$\mathbb{T}_{sc} = \{\mathbb{G}\}. \tag{5.12}$$

The social memory is updated when a position with closer distance is discovered as follows,

$$\mathbf{g}_j = \begin{cases} \mathbf{p}_{ij} & if \quad d(\mathbf{p}_{ij}, \mathbf{y}_j) < d(\mathbf{g}_j, \mathbf{y}_j) \\ \mathbf{g}_j & otherwise \end{cases}, \tag{5.13}$$

where i denotes the index of particles, j denotes the index of data vectors.

Definition 5.2.7 *(Objective Function).* The RCE minimizes a user defined objective function, $f(\mathbb{X}, \mathbb{C})$. Any internal or external cluster validity index, or any linear/product combination of multiple objective functions can be used as a possible function.

For the sake of simplicity, a generic objective function can be defined as, but not restricted to, the average distortion which is implemented as follows,

$$f_{\text{average distortion}}(\mathbb{X}, \mathbb{C}) = \frac{1}{K} \sum_{j=1}^{K} \sum_{i=1}^{N} \mathtt{u}_{ij} d(\mathbf{y}_j, \mathbf{x}_i). \tag{5.14}$$

Definition 5.2.8 *(Local Minimum).* RCE stores the **local minimum** coordinates which is a matrix of positions of non-empty particles that minimize $f(\mathbb{X}, \mathbb{C})$.

The minimum matrix returned by RCE is simply,

$$\forall t, \mathbb{X}^M = \arg\min_{\mathbf{X}} f(\mathbf{X}, \forall \mathbb{C}_{\mathbf{X}}(t) \notin \emptyset), \tag{5.15}$$

which is a set of all non-empty particles in $\mathbf{X}$ that minimizes the objective function over all iterations.

Definition 5.2.9 *(Winning Particle).* A winning particle θ_{win} is the particle which constituted voronoi region contains the most data compared to that of the rest of the particles in the swarm,

$$\theta_{win} = \arg\max_{\theta} |\mathbb{C}_{\theta}|. \tag{5.16}$$

Definition 5.2.10 *(Resultant Vector).* In the simplified PSC and RCE 2012 [4], the resultant vector $\Psi(\mathbf{x}_i) \in \mathbb{R}^{dim}$ describes the trajectory vector experienced by the i^{th} particle due to the j^{th} attractor as specified by the l^{th} term $\psi_{j\,(l)}$ in the voronoi region due to $\mathbf{x}_i$. The formulation is as

follows,

$$\begin{aligned}\Psi_{PSC}(\mathbf{x}_i) &= \sum_l \lambda_{(l)} \left(\frac{\sum_{j=1}^{N} u_{ij\,(l)}\, \varphi_{j\,(l)} \circ (\psi_{j\,(l)} - \mathbf{x}_i)}{\sum_{j=1}^{N} u_{ij\,(l)}} \right), \\ &= \sum_l \lambda_{(l)} E\left[\varphi_{(l)} | \mathbf{X} = \mathbf{x}_i\right] \circ \left(E\left[\psi_{(l)} | \mathbf{X} = \mathbf{x}_i\right] - \mathbf{x}_i \right),\end{aligned} \tag{5.17}$$

where N denotes the number of observations, l denotes the set of term functions e.g $l = \{\text{so, sc, co, } \ldots\}$, $\varphi \in \{0, 1\}$ denotes a uniform random vector in $\mathbb{R}^{dim}$, $u_{ij\,(l)} \in [0, 1]$ denotes the crisp membership of the j^{th} attractor to $\mathbf{x}_i$ due to the l^{th} term, while $\lambda_{(l)}$ denotes the corresponding coefficient for the l^{th} term.

The resultant vector is therefore the sum of the average attraction vectors imposed by each term in the corresponding voronoi region.

Analysis: There are two problems associated with Equation 5.17: The first problem is concerned with computational complexity; The second is concerned with the consequence of Kolmogorov's strong law of large number (SLLN).

The first problem is apparent from the fact that generating a pseudo-random number using algorithms such as Mersenne Twister [162] — $\mathcal{O}(p^2)$ — requires a fragment of time proportional to the number of periods $p = \log_2(N\, dim)$, where N and dim denote the volume and dimension of the dataset, respectively. Moreover this formulation would require an additional memory allocation at least as large as $\mathbb{Y}_\Theta$ to store the matrix of uniform random numbers. Obviously as the dimensionality and volume grow, generating pseudo-random numbers for Equation 5.17 would easily become a computational and memory bottleneck.

Lemma 5.2.1 *(Implications of Kolmogorov's SLLN on Ψ_{PSC}). Given a sufficiently large dataset with $N \to \infty$, Equation 5.17 converges to*

$$\lim_{N\to\infty} \Psi_{PSC}(\mathbf{x}_i) \to \sum_l 0.5\lambda_{(l)} \left(E\left[\psi_{(l)} | \mathbf{X} = \mathbf{x}_i\right] - \mathbf{x}_i \right). \tag{5.18}$$

Proof: As a consequence of Kolmogorov's SLLN, the uniform random number $\varphi \in \{0, 1\}$ converges almost surely to its expected value $E[\varphi] = 0.5$,

$$\lim_{N\to\infty} \frac{\sum_{j=1}^{N} \varphi_j}{N} \to E[\varphi] = 0.5. \tag{5.19}$$

Consequently, as $N \to \infty$,

$$E\left[\varphi_{(l)}|\mathbf{X}=\mathbf{x}_i\right] = E\left[\varphi_{(l)}\right]_{\text{over } N\to\infty \text{ trials}} \to 0.5. \tag{5.20}$$

Consequently Equation 5.17 converges to

$$\lim_{N\to\infty} \Psi_{PSC}(\mathbf{x}_i) \to \sum_l 0.5\lambda_{(l)} \left(E\left[\psi_{(l)}|\mathbf{X}=\mathbf{x}_i\right] - \mathbf{x}_i\right). \qquad \blacksquare \tag{5.21}$$

Definition 5.2.11 *(Simplified Resultant Vector)***.** The implication of Lemma 5.2.1 is severe due to the dilution of φ in larger clustering problems. Hence in order to conserve the stochastic character of Equation 5.17, we propose the following reformulation to the resultant vector computation for RCE,

$$\begin{aligned}\Psi_{RCE}(\mathbf{x}_i) &= \sum_l \lambda_{(l)}\varphi_{(l)} \circ \left(E\left[\psi_{(l)}|\mathbf{X}=\mathbf{x}_i\right] - \mathbf{x}_i\right), \text{ or even simpler,} \\ &= \varphi \circ \left[\sum_l \lambda_{(l)} \left(E\left[\psi_{(l)}|\mathbf{X}=\mathbf{x}_i\right] - \mathbf{x}_i\right)\right],\end{aligned} \tag{5.22}$$

which notations are similarly defined as Equation 5.17. The update equation is now made leaner and insensitive to $|\mathbb{Y}_\Theta|$.

Definition 5.2.12 *(Velocity)***.** The velocity vector $\mathbf{v}$ of a particle describes its movement trajectory in the Euclidean space. The velocity vector in the simplified version of PSC is updated based on the interaction of the current particle θ_i with respect to the resultant vector $\Psi_{PSC\,or\,RCE}(\mathbf{x}_i(t))$ determined by the voronoi tessellation induced by $\mathbf{X}$ as follows,

$$\mathbf{v}_i(t+1) = \begin{cases} \omega\mathbf{v}_i(t) + \Psi_{PSC\,or\,RCE}(\mathbf{x}_i(t)) & if \quad \mathbb{C}_i \neq \emptyset \\ \omega\mathbf{v}_i(t) + \varphi \circ (\mathbf{x}_{win} - \mathbf{x}_i) & otherwise \end{cases}, \tag{5.23}$$

where $\mathbf{x}_{win}$ denotes the position of the winning particle θ_{win}. The velocity is upper and lower bounded by a maximum velocity bound, which is set to a percentage $\eta\%$ of the search space Ω,

$$\mathbf{v}(t) = \max(\min(\mathbf{v}(t), \mathbf{v}_{\max}), -\mathbf{v}_{\max}), \tag{5.24}$$

$$\mathbf{v}_{max} = \eta\% \cdot \Omega. \tag{5.25}$$

Definition 5.2.13 *(Self-Organizing Vector)***.** The *self-organizing* vector $\mathbf{so}_{ij}$ describes the attraction imposed to a particle $\mathbf{x}_i$ due to the data $\mathbf{y}_j$ as follows,

$$\mathbf{so}_{ij} = \begin{cases} \mathbf{y}_j - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{5.26}$$

Definition 5.2.14 *(Social Vector).* The *social* vector $\mathbf{sc}_{ij}$ describes the attraction imposed to a particle $\mathbf{x}_i$ due to the social memory $\mathbf{g}_j$ associated with the data $\mathbf{y}_j$ as follows,

$$\mathbf{sc}_{ij} = \begin{cases} \mathbf{g}_j - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{5.27}$$

Definition 5.2.15 *(Cognitive Vector).* The *self-organizing* vector $\mathbf{co}_{ij}$ describes the attraction imposed to a particle $\mathbf{x}_i$ due to the cognitive memory $\mathbf{p}_{ij}$ of the corresponding particle $\mathbf{x}_i$ associated with the data $\mathbf{y}_j$ as follows,

$$\mathbf{co}_{ij} = \begin{cases} \mathbf{p}_{ij} - \mathbf{x}_i & if \quad \mathbf{y}_j \in \mathbb{C}_i \\ 0 & otherwise \end{cases}. \tag{5.28}$$

Definition 5.2.16 *(Minimum Vector).* The minimum vector describes the attraction imposed on $\mathbb{X}$ due to $\mathbb{X}^M$ as follows,

$$\mathbf{mi}_{ij} = \begin{cases} \mathbf{x}_j^M - \mathbf{x}_i & if \quad \mathbf{x}_i \in \mathbf{x}_j^M \\ 0 & otherwise, \end{cases} \tag{5.29}$$

where $\mathbf{x}_j^M$ denotes the local minimum vector which is closest to $\mathbf{x}_i$.

5.3 Algorithmic Fundamentals

Using the new definitions we can represent the PSC using batch matrix operation as seen in Algorithm 5.20. This new construct allows the compacting of the outer **for all** $\mathbf{y} \in \mathbb{Y}$ loop by operating on the memory matrix using Equation 5.17. The RCE proposition in 2012–2013 uses the similar construct with the addition of the local minimum term as can be seen in Algorithm 5.21.

Equation 5.22 was proposed to address Lemma 5.2.1 in RCE 2014. The social and cognitive terms were omitted in accordance to Theorem 4.1, significantly lowering the overall cost of the algorithm. The pseudocode can be seen in Algorithm 5.22.

5.4 Complexity Analysis

5.4.1 Computational Complexity

The general computational complexity of the RCE family can be summarized in Table 5.3.

Algorithm 5.20 Representing PSC using batch matrix operation

Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K, $\lambda_{(so)}, \lambda_{(co)}, \lambda_{(sc)}, \lambda_{(mi)}$.
Output: Centroid vectors $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

Initialize the swarm.
repeat
  Calculate the pairwise distance between $\mathbb{X}$ and $\mathbb{Y}$,
  Update the Social and Cognitive matrices,
  Generate $dim \times N$ random matrices: $\Phi_{so\,[dim \times N]}, \Phi_{sc\,[dim \times N]}$, and $\Phi_{co\,[dim \times N]}$,
  $\mathbf{V} \leftarrow \mathbf{V} + \Psi_{PSC}(\mathbf{x}_{1,\ldots,K})$; $\mathbb{L} = \{\text{so, sc, co}\}$,
  $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{V}$,
  Redirect particles with no member towards the winning particle.
until Convergence or maximum iteration reached
return $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

Algorithm 5.21 Proposed RCE 2012 basic algorithmic construct

Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K, $\lambda_{(so)}, \lambda_{(co)}, \lambda_{(sc)}, \lambda_{(mi)}$.
Output: Locally optimum centroid vectors $\mathbb{X}^M = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

Initialize the swarm.
repeat
  Calculate the pairwise distance between $\mathbb{X}$ and $\mathbb{Y}$,
  Update the Social and Cognitive matrices,
  Store the *minimum matrix* $\mathbb{X}^M$ which minimizes $f(\mathbb{X}, \mathbb{Y})$ (Cluster Validity),
  Generate random vectors $\varphi_{so}, \varphi_{sc}, \varphi_{co}$, and $\varphi_{mi} \in \mathbb{R}^{dim}$,
  $\mathbf{V} \leftarrow \mathbf{V} + \Psi_{PSC}(\mathbf{x}_{1,\ldots,K})$; $\mathbb{L} = \{\text{so, sc, co, mi}\}$,
  $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{V}$,
  Redirect particles with no member towards the winning particle.
until Convergence or maximum iteration reached
return $\mathbb{X}^M = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

Algorithm 5.22 Proposed RCE 2014 basic algorithmic construct

Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K, $\lambda_{(so)}, \lambda_{(mi)}$.
Output: Locally optimum centroid vectors $\mathbb{X}^M = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

Initialize the swarm.
$\mathbb{Y}_\Theta = randsample(\mathbb{Y}, \eta\%)$.
repeat
  Calculate the pairwise distance between $\mathbb{X}$ and $\mathbb{Y}_\Theta$,
  Store the *minimum matrix* $\mathbb{X}^M$ which minimizes $f(\mathbb{X}, \mathbb{Y}_\Theta)$ (Cluster Validity),
  Generate random vectors φ_{so}, and $\varphi_{mi} \in \mathbb{R}^{dim}$,
  $\mathbf{V} \leftarrow \mathbf{V} + \Psi_{RCE}(\mathbf{x}_{1,\ldots,K})$; $\mathbb{L} = \{\text{so, mi}\}$,
  $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{V}$,
  Redirect particles with no member towards the winning particle.
until Convergence or maximum iteration reached
return $\mathbb{X}^M = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$.

TABLE 5.3: Worst case computational complexity of the RCE families vs PSC

Task	**PSC**	**RCE 2012**	**RCE2014**
Pairwise distance	$\mathcal{O}(NKdim)_{\textbf{for loop}}$	$\mathcal{O}(NKdim + K^2dim)_{\textbf{matrix}}$	
comments	On each iteration, the PSC uses a serial for loop to calculate $d(\mathbf{y}, \mathbb{X}), \forall \mathbf{y} \in \mathbb{Y}$.	On each iteration, the RCE utilizes fully vectorized matrix computation to calculate the $K \times N$ distance matrix $D(\mathbb{Y}, \mathbb{X})$ and the $K \times K$ distance matrix $D(\mathbb{X}^M, \mathbb{X})$	
Φ or φ generation	$\mathcal{O}(\log_2(3(N+K)dim))$	$\mathcal{O}(\log_2((3N + K^2 + K)dim))$	$\mathcal{O}(\log_2 2Kdim)$
comments	Each iteration the PSC needs to generate a $3 \times N \times dim$ matrix of random numbers for {so,sc,co}, and an additional $K \times dim$ matrix considering the worst case scenario where there are $K-1 \approx K$ particles with empty voronoi regions.	Each iteration RCE 2012 needs to generate at most three matrices of random numbers: a $3 \times N \times dim$ matrix for {so,sc,co}; a $K \times K \times dim$ matrix for {mi}, and; a $K \times dim$ matrix for $K - 1 \approx K$ particles with empty voronoi regions.	Each iteration RCE 2014 needs to generate at most two dim-dimensional vectors of random numbers for each particles: two vectors for {so, mi} if its voronoi region is nonempty; and one vector only if its voronoi region is empty.
V and **X** update	$\mathcal{O}(16(N+K)dim)$	$\mathcal{O}(3Ndim + Kdim)$	$\mathcal{O}(Ndim + Kdim)$
comments	The PSC needs to update the position of each particle on each encounter with a data point and when the voronoi region is empty. The update is done serially inside a for loop. The cost of a position update is approximately $\mathcal{O}(16dim)$ based on the number of floating point operations.	The RCE 2012 update scheme requires at most $3 \times N \times dim$ floating point operations for calculating the resultant vector for {so,sc,co}, and; $K \times dim$ floating point operations to calculate the resultant vector for {mi}.	The RCE 2014 update scheme requires at most $N \times dim$ floating point operations for calculating the resultant vector for {so}, and; $K \times dim$ floating point operations to calculate the resultant vector for {mi}.
memory update	$\mathcal{O}(NK + 4N + 2Ndim)$	$\mathcal{O}(NK + 4N + 2Ndim + K + Kdim)$	$\mathcal{O}(K + Kdim)$
comments	On each iteration, the computational complexity for updating the memory matrices $\mathbb{G}$ and $\mathbb{P}$	On each iteration, the computational complexity for updating the memory matrices $\mathbb{G}$, $\mathbb{P}$, and $\mathbb{X}^M$	On each iteration, the computational complexity for updating the local minimum matrix $\mathbb{X}^M$.

5.4.2 Memory Complexity

The memory complexity of the PSC and RCE depends on the term that it uses. The complete breakdown of the memory complexities has been listed in Table 4.1 in the previous chapter. An empirical observation of the memory allocation of each algorithm during runtime has been previously presented in Figure 4.3. The aforementioned Table and Figure are again reproduced in Table 5.4 and Figure 5.3 for the sake of clarity.

As can be seen in Figure 5.3 and Table 5.4, the algorithms of the PSC families are considerably expensive in terms of memory complexity. Our benchmark test revealed that in order to cluster 1000 observations of 1000 dimensional double precision data, the PSC families and RCE 2012 require as much as 2GB memory; whereas k-means, fuzzy c-means, and RCE^{r+} 2014 require memory allocation of lower than 15MB. This relatively high memory requirement can be a significant scalability bottleneck when dealing with larger, higher dimensional datasets.

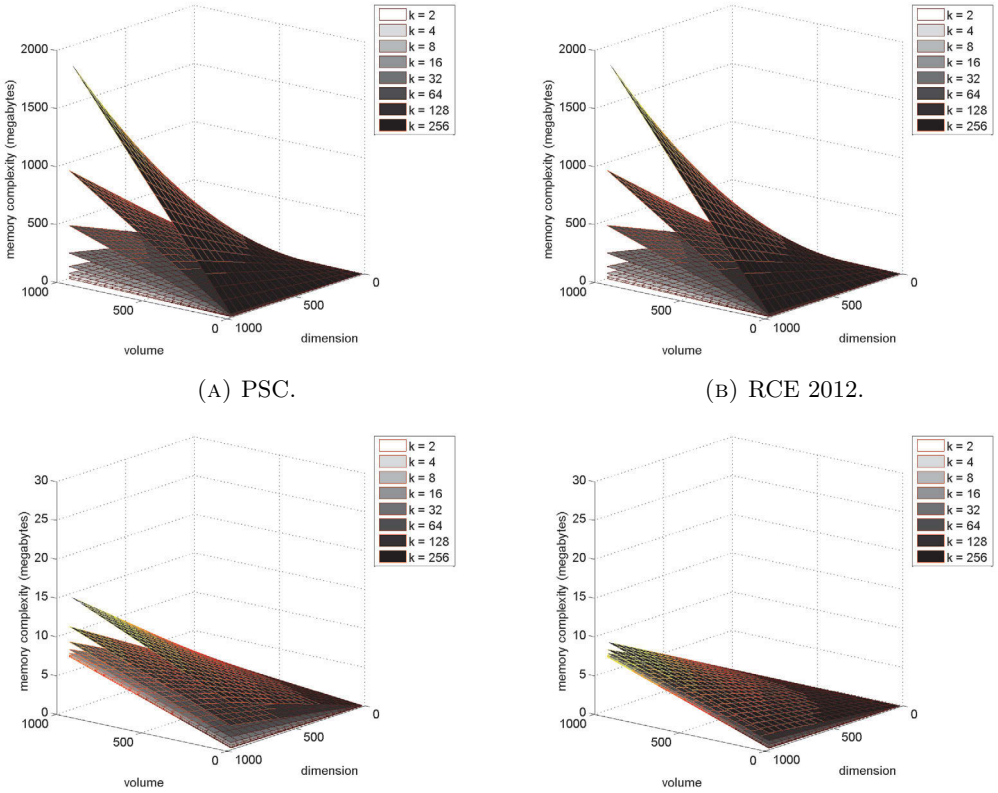
TABLE 5.4: Memory complexity of the RCE, PSC, and other clustering algorithms

Entity	Complexity	Fuzzy c-means	k-means	PSC [75]	mPSC [74]	FPSC [70]	RCE^{r+} (2012) [4]	RCE^{r+} (2014) [55]	Description
$\mathbf{Y}$	$\mathcal{O}(Ndim)$	✓	✓	✓	✓	✓	✓	✓	The data vectors: $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$
$\mathbf{X}$	$\mathcal{O}(Kdim)$	✓	✓	✓	✓	✓	✓	✓	The particle position vectors: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in \mathbb{R}^{dim}$
$\mathbf{V}$	$\mathcal{O}(Kdim)$			✓		✓	✓	✓	The particle velocity vectors: $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\} \in \mathbb{R}^{dim}$
$\mathbf{v}_{max}$	$\mathcal{O}(dim)$			✓		✓	✓	✓	The maximum velocity vector: $\mathbf{v}_{max} \in \mathbb{R}^{dim}$
U	$\mathcal{O}(N)$		✓	✓	✓		✓	✓	The crisp label vector or binary indicator matrix: $\mathsf{U} = \{\mathsf{u}_1, \ldots, \mathsf{u}_N\} \in \mathbb{R}^1$
$\mathbf{U}$	$\mathcal{O}(NK)$	✓				✓		✓	The fuzzy membership matrix: $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\} \in \mathbb{R}^K$
$\mathbf{X}_{best}$	$\mathcal{O}(Kdim)$						✓	✓	The best position matrix: $\mathbf{X}_{best} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}_{best} \in \mathbb{R}^{dim}$
$f(\mathbb{C}_{Xbest}, \mathbf{Y})$	$\mathcal{O}(1)$						✓	✓	The quality of the voronoi tessellation imposed by $\mathbf{X}_{best}$
$\mathbf{G}$	$\mathcal{O}(Ndim)$			✓	✓	✓	✓		The swarm social memory vector for each data: $\mathbf{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_N\} \in \mathbb{R}^{dim}$
$d(\mathbf{g}_j, \mathbf{y}_j)_{\{\forall i, \forall j\}}$	$\mathcal{O}(N)$			✓	✓	✓	✓		The distance between the vectors in the social memory relative to its corresponding data vector, $d(\mathbf{p}_{i,j}, \mathbf{y}_j) \in \mathbb{R}^1$
$\mathbf{P}_{\{1,\ldots,K\}}$	$\mathcal{O}(NKdim)$			✓	✓	✓	✓		The cognitive memory for each particle, for each data: $\mathbf{P}_i = \{\mathbf{p}_{i,1}, \ldots, \mathbf{p}_{i,N}\} \in \mathbb{R}^{dim}$
$d(\mathbf{p}_{i,j}, \mathbf{y}_j)_{\{\forall i, \forall j\}}$	$\mathcal{O}(NK)$			✓	✓	✓	✓		For each particle, the relative distance between the cognitive memory vector and its corresponding data vector, $d(\mathbf{p}_{i,j}, \mathbf{y}_j) \in \mathbb{R}^1$

5.4.3 Empirical Experiment

A three classes 80-dimensional Gaussian distributed data of equal variance and varying volumes were generated in order to benchmark the memory and computational cost of each algorithm. The algorithmic complexities are presented as bar graphs in Figure 5.4 for ease of interpretation.

The experimental results confirms the theoretical complexity analyses done in the previous subsections. As expected, the simplification scheme for the PSC leads to increased time efficiency at the expense of memory complexity. We validated that RCE 2014 achieved the lowest complexity in

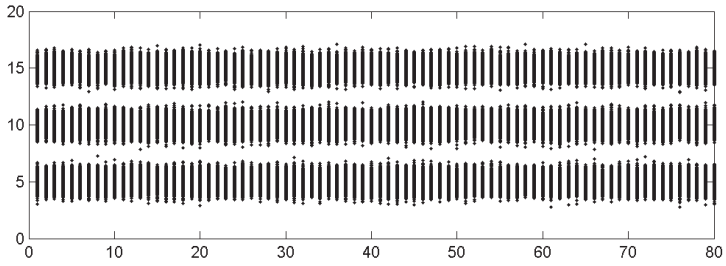


(A) PSC. (B) RCE 2012.

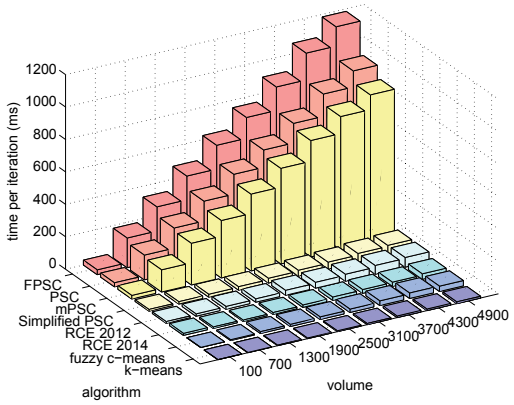
(C) RCE^{r+} (2014). (D) k-means.

FIGURE 5.3: Memory complexity of various algorithms when clustering double precision floating point numbers. The runtime test is carried using Matlab.

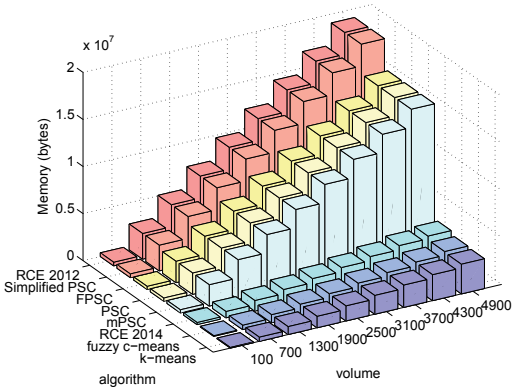
both time and space where its overall memory and computational complexity sit closely to those of k-means and fuzzy c-means. Contrary to the belief where higher cost tends to reflect higher quality, the relatively low cost of RCE 2014 translates to an increase in repeatability and clustering quality as shown previously in Figure 5.2.



(A) The scatter plot of the 80 dimensional, 3-classes dataset used for the benchmarking process.



(B) Computational complexity, low to high.



(C) Memory complexity, low to high.

FIGURE 5.4: Benchmarking the iteration complexity and overall memory complexity of various algorithm using a 3-classes 80-dimensional Gaussian dataset.

5.5 Trajectory Analysis

5.5.1 Stability and Convergence

Theorem 5.1 (RCE's resemblance to k-means)**.** *Under the condition where* $\sum_l \lambda_{(l)}$ *obeys the stability constraints,*

$$0 \leq \omega < 1, \text{ and } 0 < \sum_l \lambda_{(l)} < 2 + 2\omega, \tag{5.30}$$

each particle would converge to

$$E[x(t \to \infty)] = \frac{\sum_l \lambda_{(l)} E\left[\psi_{(l)} | X = x(t \to \infty)\right]}{\sum_l \lambda_{(l)}}. \tag{5.31}$$

A monotonic decrease in the distortion function is guaranteed when the self organizing term is enabled. The RCE generalizes to k-means or fuzzy c-means when the algorithm uses only the self organizing term.

Proof: The overall position update equation of RCE, seen from a specific particle and dimension, can be summarized as follows,

$$v(t+1) = \omega v(t) + \Psi(x(t)) \tag{5.32}$$

$$= \omega v(t) + \varphi \sum_l \lambda_{(l)} \left(E\left[\psi_{(l)} | X = x(t)\right] - x(t) \right), \tag{5.33}$$

$$= -\varphi \sum_l \lambda_{(l)} x(t) + \omega v(t) + \varphi \sum_l \lambda_{(l)} E\left[\psi_{(l)} | X = x(t)\right], \tag{5.34}$$

$$x(t+1) = x(t) + v(t+1), \tag{5.35}$$

$$= \left(1 - \varphi \sum_l \lambda_{(l)}\right) x(t) + \omega v(t) + \varphi \sum_l \lambda_{(l)} E\left[\psi_{(l)} | X = x(t)\right]. \tag{5.36}$$

Based on the analysis in Lemma 3.1.1 we represent φ using the supremum of the set such that $\sup(\varphi) = 1$. Let $\mathbf{X} = [x(t), v(t)]^T$ and $\mathbf{U}(t) = \left[E\left[\psi_{(1)} | X = x(t)\right], E\left[\psi_{(2)} | X = x(t)\right], \ldots\right]^T$, using the supremum we can express the above equation in an explicit time-invariant state space format,

$$\begin{bmatrix} x(t+1) \\ v(t+1) \end{bmatrix} = \overbrace{\begin{bmatrix} 1 - \sum_l \lambda_{(l)} & \omega \\ -\sum_l \lambda_{(l)} & \omega \end{bmatrix}}^{\sup(\mathbf{A})} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \overbrace{\begin{bmatrix} \lambda_{(1)} & \lambda_{(2)} & \cdots \\ \lambda_{(1)} & \lambda_{(2)} & \cdots \end{bmatrix}}^{\sup(\mathbf{B})} \begin{bmatrix} E\left[\psi_{(1)} | X = x(t)\right] \\ E\left[\psi_{(2)} | X = x(t)\right] \\ \vdots \end{bmatrix} \tag{5.37}$$

$$\mathbf{Y}(t) = \overbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}^{\mathbf{C}} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix}. \tag{5.38}$$

The transfer function is consequently,

$$\mathbf{H}(z) = \frac{1}{z^2 + \left(\sum_l \lambda_{(l)} - \omega - 1\right) z + \omega} \begin{bmatrix} \lambda_{(1)} z & \lambda_{(2)} z & \cdots \end{bmatrix}. \tag{5.39}$$

Using the approach in Lemma 3.1.1, we have a guaranteed convergence when

$$0 \leq \omega < 1, \text{ and } 0 < \sum_l \lambda_{(l)} < 2 + 2\omega. \tag{5.40}$$

Applying the final value theorem [158] to the RCE transfer function, we obtain convergence towards the arithmetic center of the expected values conditional on the convergence position $x(t \to \infty)$,

$$x(t \to \infty) = \lim_{z \to 1} \left\{ (z-1) \mathbf{H}(z) \mathbf{U}(z) \right\}, \tag{5.41}$$

$$= \frac{\sum_l \lambda_{(l)} E\left[\psi_{(l)} | X = x(t \to \infty)\right]}{\sum_l \lambda_{(l)}} \qquad \square \tag{5.42}$$

It then follows naturally that the RCE generalizes to the k-means / fuzzy c-means simply by enabling only the self-organizing term $\mathbb{L} = \{so\}$ and setting $\lambda_{(so)} < 2 + 2\omega$ as follows,

$$x(t \to \infty) = \frac{\lambda_{(so)} E\left[\psi_{(so)} | X = x(t \to \infty)\right]}{\lambda_{(so)}} \tag{5.43}$$

$$= E\left[Y | X = x(t \to \infty)\right], \tag{5.44}$$

$$= \frac{1}{|\mathbb{C}_x|} \sum_{y \in \mathbb{C}_x} y, \qquad \textit{k-means} \tag{5.45}$$

$$= \frac{\sum_j u_{ij} y_j}{\sum_j u_{ij}}, \qquad \textit{fuzzy c-means / EM-GMM} \tag{5.46}$$

which therefore guarantees the convergence of RCE to a local minimum given a specified initial position $x(t = 0)$. ■

With the same spirit, we can then compute the guaranteed point of convergence for RCE 2014. Enabling both self-organizing and minimum term $\mathbb{L} = \{so, mi\}$ and setting $\lambda_{(so)} + \lambda_{(mi)} < 2 + 2\omega$ gives,

$$x(t \to \infty) = \frac{\lambda_{(so)} E\left[Y | X = x(t \to \infty)\right] + \lambda_{(mi)} E\left[X^M | X = x(t \to \infty)\right]}{\lambda_{(so)} + \lambda_{(mi)}}, \tag{5.47}$$

which is the weighted arithmetic center between the local minimum and the maximum likelihood (self-organizing) coordinates at convergence.

An illustration based on Game Theory: This behavior can be illustrated and simplified by borrowing the concepts from Game Theory as follows. Let us assume an all-knowing oracle appears to the player at the current time t. The oracle ensures the player that the trajectory $\Psi_{(so)}(\mathbf{x}_i)$ leads to the maximum likelihood estimate given the current cluster assignment. The player listens to the oracle, and proceed with a velocity vector imparted from the knowledge of both the oracle $\Psi_{(so)}$ and the swarm $\Psi_{(mi)}$ weighted by $\lambda_{(so)}$ and $\lambda_{(mi)}$. Such situation is illustrated as follows.

Oracle: Let $\mathbf{z}_i^* \leftarrow mean(\mathbb{C}_i)$, then $\mathbf{z}_i^* = \mathbf{x}_i + \Psi_i^*$ where $\Psi_i^* = \Psi_{(so)}(\mathbf{x}_i)$. The oracle gives the player $\Psi^*(\mathbf{x}_i)$.

Player (particle): Listens to the oracle and proceed with $\hat{\mathbf{x}}_i = \mathbf{x}_i + \Psi_i^\dagger$, where $\Psi_i^\dagger = \omega \dot{\mathbf{v}}_i + \Psi_{(so)}(\mathbf{x}_i) + \Psi_{(mi)}(\mathbf{x}_i)$, an accumulative augmented knowledge from both the oracle and the swarm.

The player's movement trajectory can be decomposed into three additive vectors: $\omega \dot{\mathbf{v}}_i$, $\Psi_{(so)}$ and $\Psi_{(mi)}$. The cosine angle between each vector with the oracle's (self-organizing) vector $\Psi_i^* = \Psi_{(so)}$

are

$$cos(\omega\dot{\mathbf{v}}_i, \Psi_i^*) = \frac{\langle \omega\dot{\mathbf{v}}_i, \Psi_{(so)} \rangle}{|\omega\dot{\mathbf{v}}_i||\Psi_{(so)}|}, \tag{5.48}$$

$$cos(\Psi_{(so)}, \Psi_i^*) = \frac{\langle \Psi_{(so)}, \Psi_{(so)} \rangle}{|\Psi_{(so)}||\Psi_{(so)}|} = 1, \tag{5.49}$$

$$cos(\Psi_{(mi)}, \Psi_i^*) = \frac{\langle \Psi_{(mi)}, \Psi_{(so)} \rangle}{|\Psi_{(mi)}||\Psi_{(so)}|}, \tag{5.50}$$

where obviously one of the components is guaranteed to perfectly align with Ψ_i^*. The local minimum vector will be aligned with the self-organizing vector if the path to the swarm's local minimum is also the self-organizing vector. If the minimum vector is not aligned with the self-organizing vector, then there is a possibility for the swarm to instead traverse the path of the local minimum.

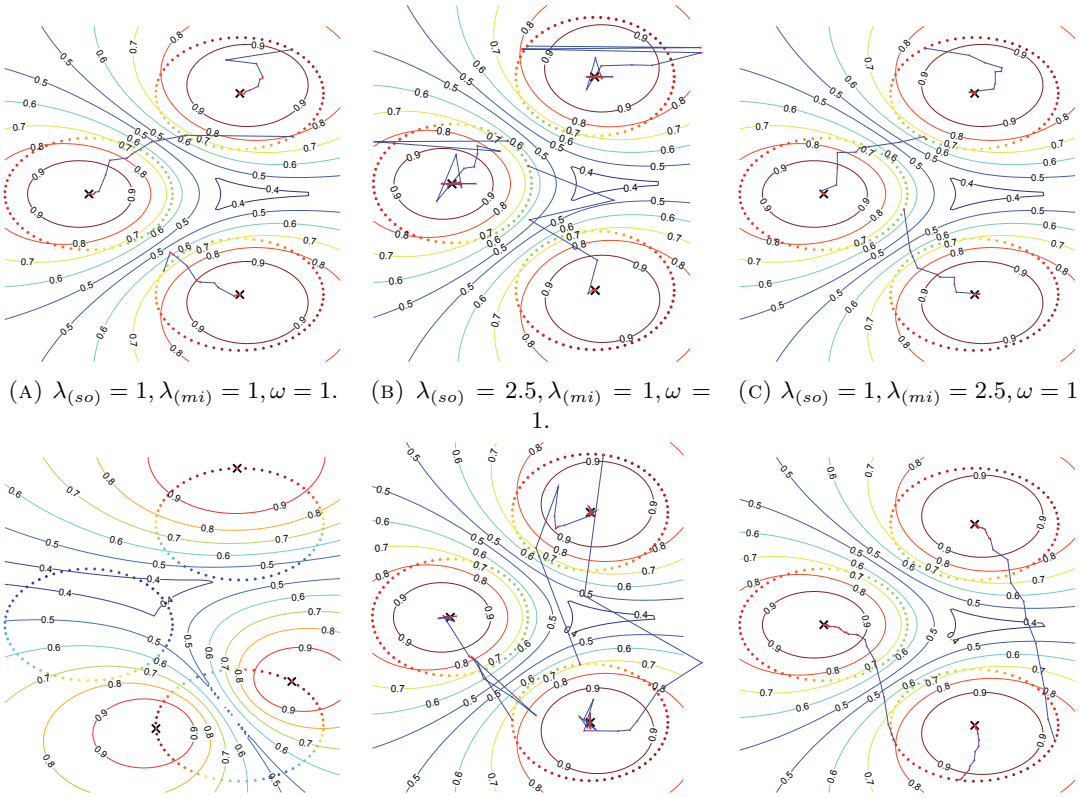
5.5.2 Particle Behavior

We observe the trajectory using the same datasets as the PSC: namely the three circle dataset, and; the five Gaussian dataset. The results can be seen in Figure 5.5 and Figure 5.6. Its trajectory resembles the PSC using the updated stability bounds in Equation 4.37. Notice that higher value in the self-organizing coefficient results in numerous overshoots, thus more abrupt particle movements. From these results one can easily observe that the RCE is also dependent on initial position as proven in Theorem 5.1.

5.6 Coping with Local Optima

At a glance, the particle behavior of the RCE resembles the k-means or fuzzy c-means. However, the RCE benefit from the minimum term and the exploratory capability by overshooting the maximum likelihood vector. The RCE is therefore not fully stochastic, because its movement trajectory is directly "suggested" by the maximum likelihood vector. We can afford to do this because clustering is different from normal optimization where the first derivative of the likelihood function is known.

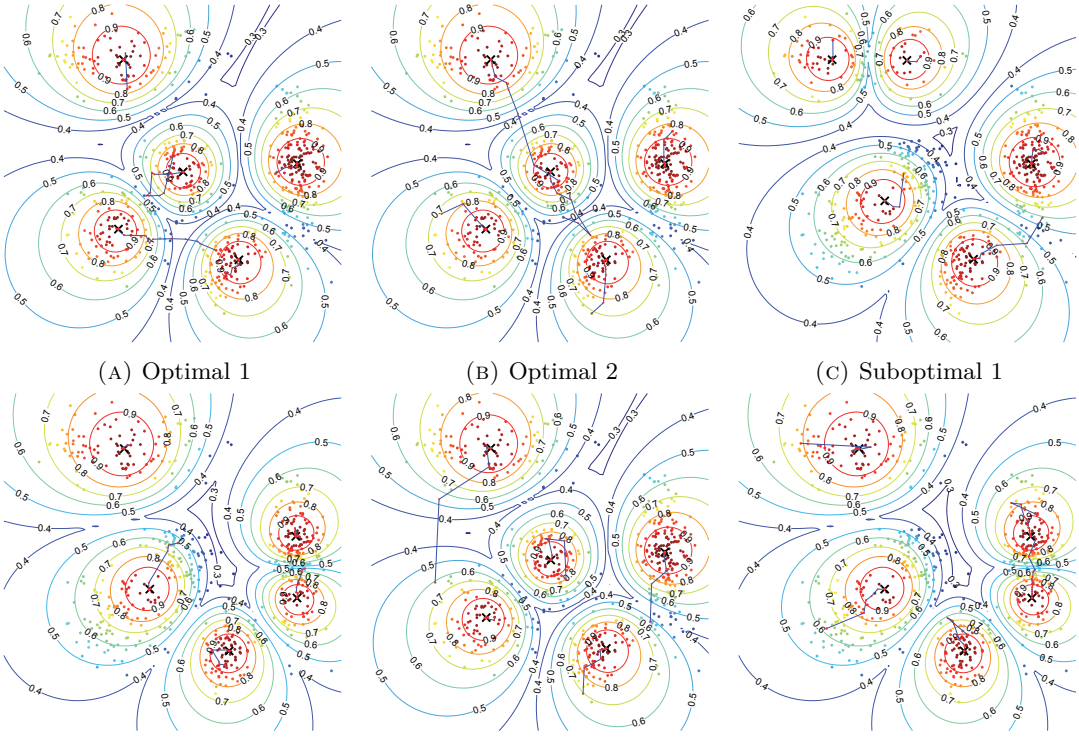
We acknowledge that there are numerous local optima and that there is one unique global optimum which can be quantified using the appropriate objective function. Accordingly, we realize that settling at any "equilibrium" would increase the risk of suboptimal convergence. Therefore, we propose strategies including *substitution*, *particle reset*, and *multi-swarm* to gracefully break these equilibriums.



(A) $\lambda_{(so)} = 1, \lambda_{(mi)} = 1, \omega = 1.$ (B) $\lambda_{(so)} = 2.5, \lambda_{(mi)} = 1, \omega = 1.$ (C) $\lambda_{(so)} = 1, \lambda_{(mi)} = 2.5, \omega = 1$

(D) $\lambda_{(so)} = 0, \lambda_{(mi)} = 3.5, \omega = 1$, does not converge. (E) $\lambda_{(so)} = 3.5, \lambda_{(mi)} = 0, \omega = 1.$ (F) $\lambda_{(so)} = 0.5, \lambda_{(mi)} = 0.5, \omega = 1$

FIGURE 5.5: Trajectory of the RCE 2014 particles using various parameters.



(A) Optimal 1 (B) Optimal 2 (C) Suboptimal 1

(D) Suboptimal 2 (E) Optimal 3 (F) Suboptimal 3

FIGURE 5.6: Trajectory of the RCE 2014 particles on the five Gaussian dataset with numerous random seeding shows RCE's sensitivity to initialization.

5.6.1 Substitution

The purpose of the *Substitution* strategy is to force particles in a search space to reach alternate equilibrium positions by introducing position instability. After each position update episode for a particle, apply

$$\begin{aligned} \mathbf{x}_i(t+1) &= \begin{cases} \mathbf{x}_{I\,win}(t+1) + \mathbf{N}(0,\sigma) & \text{if } \varphi < \varepsilon \\ \mathbf{x}_i(t+1) & \text{otherwise} \end{cases}, \\ \Delta\mathbf{x}_i(t+1) &= \begin{cases} 0 & \text{if} \quad \varphi < \varepsilon \\ \Delta\mathbf{x}_i(t+1) & \text{otherwise} \end{cases}, \end{aligned} \tag{5.51}$$

where φ is a uniform random number $0 \leq \varphi \leq 1 \in \mathbb{R}$, $\mathbf{x}_{I\,win}$ is the position of the winning particle, and $\mathbf{N}(0,\sigma)$ is a Gaussian random vector with mean $\mu = 0$ and standard deviation σ of each dimension of the data being clustered. ε denotes the *substitution* probability parameter. Larger ε increases the frequency. Optimal ε values lie between $0.01 \leq \varepsilon \leq 0.05$ [4].

The *substitution* strategy can be viewed analogous to the mutation operator in the Genetic Algorithm (GA), where the mutative gene is taken from the coordinate of the winning particle. ε quantifies the mutation probability, e.g. $\varepsilon = 0.05$ denotes that for each iteration, each particles have 5% chance of entering the *substitution* mode, or in the context of GA – mutate.

The superscript plus (+) (e.g. RCE$^+$) denotes an RCE with *Substitution* strategy.

5.6.2 Particle Reset

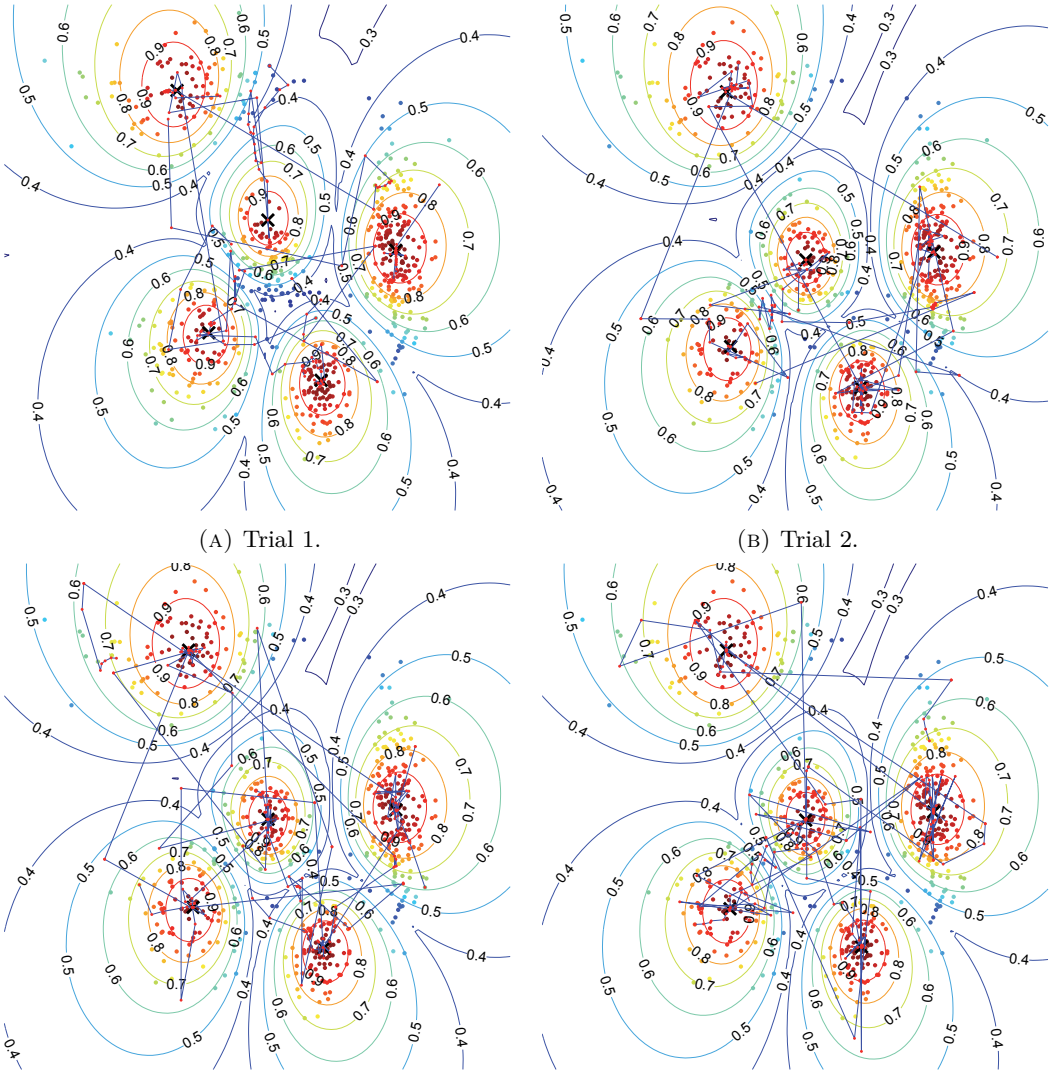
The *Particle Reset* strategy is triggered when fitness of the local minimum $f(\mathbf{X}^M(t), \mathbf{y})$ does not improve after a number of iterations. Stagnation can be detected using a stagnation counter δ which is updated as follows:

$$\delta(t+1) = \begin{cases} \delta(t) + 1 & \text{if} \quad f(\mathbf{x}(t), \mathbf{y}) \geq f(\mathbf{X}^M(t), \mathbf{y}) \\ 0 & \text{if} \quad f(\mathbf{x}(t), \mathbf{y}) < f(\mathbf{X}^M(t), \mathbf{y}) \end{cases}. \tag{5.52}$$

When $\delta(t+1) > \delta_{max}$ this strategy reinitializes all particles in a subswarm without resetting the local minimum position matrix $\mathbf{X}^M(t)$. Values being reinitialized are only $\mathbf{x}_i(t)$ and $\Delta\mathbf{x}_i(t)$. Swarm convergence is detected when $f(\mathbf{X}^M(t), \mathbf{y})$ does not improve even after many resets.

The superscript r (e.g. RCE^r) denotes an RCE with *Particle Reset* strategy.

Using both the *substitution* and *particle reset* strategies allows the particles to continuously search for a global minimum regardless of numerous suboptimal convergence. The two strategies therefore reduce the likelihood of getting trapped inside local optima. RCE^{r+} is therefore less sensitive to initialization and is almost guaranteed to find the natural global optimum clustering solution given enough iterations. The particle trajectory of RCE^{r+} on the five Gaussian dataset can be seen in Figure 5.7.



(A) Trial 1. (B) Trial 2.

(C) Trial 3. (D) Trial 4.

FIGURE 5.7: Trajectory of the RCE^{r+} 2014 particles on the five Gaussian dataset with numerous random seeding using both *substitution* and *particle reset* strategies reflects relative robustness and insensitivity to initialization. $\lambda_{(so)} = 1.5, \lambda_{(mi)} = 1.5, \omega = 1, t_{max} = 30, \varepsilon = 0.05, \delta_{max} = 15$.

5.6.3 Swarm RCE: The Multi-Swarm Paradigm

A limitation inherited from PSC is that the number of particles in a swarm is fixed according to the desired number of clusters. To overcome this limitation, a strategy is proposed that is intended to handle increases in swarm size, without increasing the number of clusters [72]. A subswarm, Θ, consists of K particles, each corresponding to a cluster centroid prototype.

A *Swarm*$\{n_m\}$ RCE consists of n_m RCE subswarms working in parallel. For example, *Swarm*$\{3\}$ RCE indicates a centroid optimization using 3 RCE subswarms, while *Swarm*$\{5\}$ RCE indicates a centroid optimization using 5 RCE subswarms.

Each RCE subswarm RCE{n} stores a best position matrix $\mathbf{X}_n^M(t)$. The *swarm* strategy communicates each $\mathbf{X}^M(t)$ such that the potentially optimal positions are informed to the subswarms.

On the start of every iteration, each subswarm contributes by sharing its minimum matrix $\mathbf{X}_n^M(t)$ such that

$$\mathcal{X}^M(t) = \left\{[\mathbf{X}_1^M], [\mathbf{X}_2^M], \ldots, [\mathbf{X}_{n_m}^M]\right\} \tag{5.53}$$

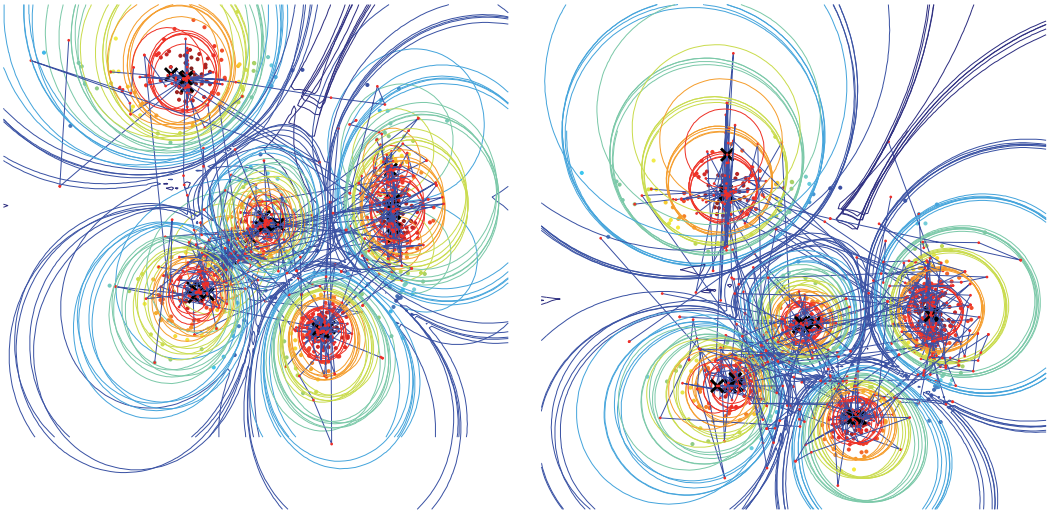
The matrix $\mathcal{X}^M$ has $K \times n_m$ columns denoting the number of centroid vectors stored in $\mathcal{X}^M$. When using the *Swarm* strategy, the $\Psi_{(mi)}$ uses $\mathcal{X}^M$ instead of the individual $\mathbf{X}_n^M$.

The pseudocode for *Swarm*$\{n_m\}$ RCE^{r+} can be seen in Algorithm 5.23.

Algorithm 5.23 *Swarm*$\{n_m\}$ RCE^{r+} 2014

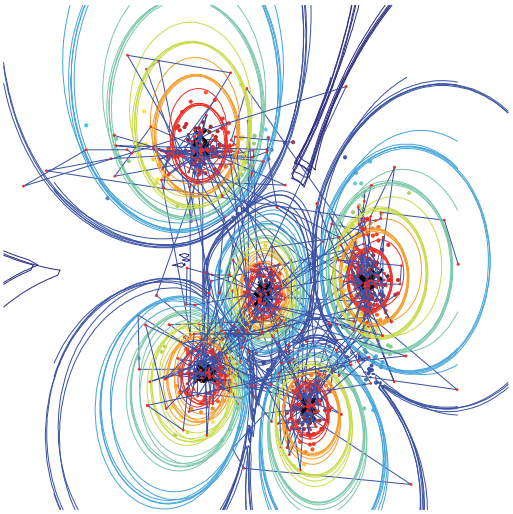
Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of clusters K, # of swarms n_m, $\lambda_{(so)}, \lambda_{(mi)}$, maximum stagnation δ_{max}, substitution rate ε.
Output: The swarm global optimum $\mathcal{X}^M$ and the corresponding cluster validity $f(\mathcal{X}^M, \mathbb{Y})$.
1: Initialize swarm $\Theta_{1,\ldots,n_m}$.
2: $(\forall \Theta \in \mathbb{S}), \mathbb{Y}_\Theta = randsample(\mathbb{Y}, \eta\%)$.
3: **repeat**
4: **for** $\mathbb{S} = \{\Theta_1, \ldots, \Theta_{n_m}\}$
5: Update the swarm Θ [Algorithm 5.22 lines 4–9].
6: Apply substitution at rate of ε
7: **if** $f(\mathbb{X}_\Theta, \mathbb{Y}_\Theta)$ does not improve after δ_{max} iterations
8: Apply particle reset,
9: Reset stagnation counter δ
10: **end if**
11: **end for**
12: **until** Convergence ($f(\mathcal{X}^M, \mathbb{Y}_\Theta)$ does not improve after n_{max} resets) or maximum iteration reached
13: **return** $\mathcal{X}^M = \{[\mathbf{X}_1^M], \ldots, [\mathbf{X}_{n_m}^M]\} \in \mathbb{R}^{dim}$ and $f(\mathcal{X}^M, \mathbb{Y}) = \{f(\mathbb{X}_1^M, \mathbb{Y}), \ldots, f(\mathbb{X}_{n_m}^M, \mathbb{Y})\} \in \mathbb{R}^{dim}$.

The particle trajectory of *Swarm*$\{6\}$ RCE^{r+} on five Gaussian dataset can be seen in Figure 5.8. In this figure, it can be seen that in just 30 iterations the swarm has already explored a considerable amount of optimum. The multi-swarm strategy alleviates RCE's sensitivity to initialization.

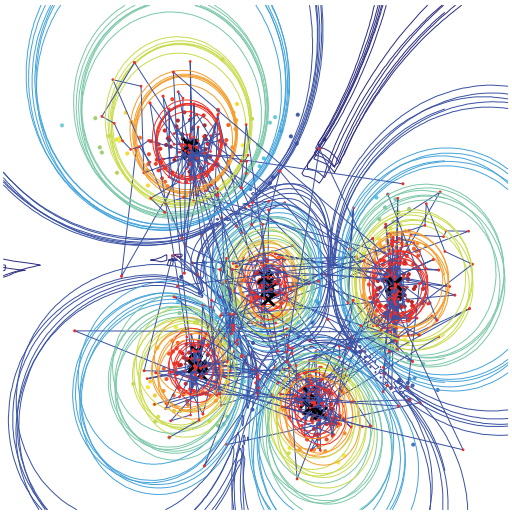


(A) Trial 1.

(B) Trial 2.



(C) Trial 3.



(D) Trial 4.

FIGURE 5.8: Trajectory of the *swarm*{6} RCE^{r+} 2014 particles recorded after 30 iterations on the five Gaussian dataset with numerous random seeding using both *substitution* and *particle reset* strategies shows *Swarm*{6} RCE^{r+} robustness and insensitivity to initialization. $\lambda_{(so)} = 1.5, \lambda_{(mi)} = 1.5, \omega = 1$, $n_m = 6$, $t_{max} = 30$, $\varepsilon = 0.05$, $\delta_{max} = 15$.

5.7 Experimental Results

The comparative results of the algorithm on some of the datasets available in UCI machine learning dataset repository [84] according to our 2012 publication [4] can be seen in Table 5.5. Note that the RCE algorithms used in this table are based on RCE 2012 [4, 71–73], prior to the 2014 simplification [55].

The results given in Table 5.5 indicate that the performance of the RCE 2012 algorithms was superior to that of the predecessors. It can be seen that when using both *Substitution* and *Particle*

TABLE 5.5: Performance against UCI Machine Learning datasets [84] as reported in [4].

Dataset		**Algorithm**					
	Performance Metric	K-means	PSC	mPSC	RCE 2012	RCE^{r+} 2012	*Swarm*{5} RCE^{r+} 2012
Iris (Correlation)	Classification Entropy	0.16 ± 0.03	0.31 ± 0.106	0.32 ± 0.090	0.16 ± 0.173	0.16 ± 0.173	**0.15 ± 0.02**
	Purity	78.6% ± 16.8%	79.9% ± 11.6%	88.6% ± 10.7%	91.9% ± 8%	94.61% ± 4.9%	**95.8% ± 0.66%**
	Time required (s)	**5.4e-3 ± 0.8e-3**	6.87 ± 8.51	6.39 ± 8.39	0.19 ± 0.136	0.32 ± 0.130	2.13 ± 0.93
	Dunn Index	5.1 ± 2.58	5.01 ± 2.37	5.69 ± 1.89	4.55 ± 2.36	6.44 ± 0.41	**6.54 ± 0.218**
	Adj. Rand Index	0.78 ± 0.19	0.76 ± 0.15	0.82 ± 0.13	0.75 ± 0.16	0.87 ± 0.04	**0.88 ± 0.02**
	CH Index	2.7e4 ± 1.1e4	2.3e4 ± 1.2e4	2.8e4 ± 0.9e4	2.3e4 ± 1.2e4	3.2e4 ± 0.1e4	**3.3e4 ± 0.03e4**
Glass (Correlation)	Classification Entropy	0.91 ± 0.112	0.626 ± 0.184	**0.584 ± 0.154**	0.679 ± 0.164	0.75 ± 0.15	0.797 ± 0.110
	Purity	48.0% ± 6.25%	47.7% ± 5.1%	46.2% ± 5.9%	49.1% ± 5.7%	50.3% ± 4.9%	**51.5% ± 4.8%**
	Time required (s)	**5.6e-3 ± 2.2e-3**	9.3 ± 8.03	7.84 ± 7.55	0.3 ± 0.098	0.25 ± 0.08	1.65 ± 0.32
	Dunn Index	0.3 ± 0.25	0.39 ± 0.32	0.41 ± 0.27	0.45 ± 0.56	0.45 ± 0.31	**0.47 ± 0.29**
	Adj. Rand Index	0.2 ± 0.08	**0.5 ± 0.09**	0.5 ± 0.11	0.48 ± 0.19	0.32 ± 0.12	0.31 ± 0.1
	CH Index	1.9e3 ± 1.2e3	3.3e3 ± 1.5e3	**3.5e3 ± 1.3e3**	1.6e3 ± 0.6e3	2.7e3 ± 1.0e3	3.0e3 ± 1.2e3
Wine (Correlation)	Classification Entropy	0.61 ± 0.03	0.1806 ± 0.05	0.189 ± 0.06	0.18 ± 0.027	0.18 ± 0.026	**0.18 ± 0.015**
	Purity	66.7% ± 7.16%	79.38% ± 13.8%	82.7% ± 14.1%	90.6% ± 8.7%	95.2% ± 0.87%	**95.31% ± 0.53%**
	Time required (s)	**6.2e-3 ± 2.1e-3**	4.4 ± 6.13	3.54 ± 5.14	0.27 ± 0.11	0.35 ± 0.09	2.19 ± 0.53
	Dunn Index	3.36 ± 0.009	2.25 ± 0.74	2.46 ± 0.84	2.47 ± 1.14	3.32 ± 0.143	**3.38 ± 0.06**
	Adj. Rand Index	0.316 ± 0.03	0.314 ± 0.03	0.291 ± 0.05	0.306 ± 0.04	0.322 ± 0.02	**0.322 ± 0.01**
	CH Index	4.2e3 ± 0.01e3	2.65e3 ± 1.5e3	2.4e3 ± 1.2e3	2.8e3 ± 1.7e3	4.15e3 ± 0.3e3	**4.21e3 ± 0.09e3**
Breast Cancer (Euclidean)	Classification Entropy	0.17 ± 0.0	**0.16 ± 0.014**	0.17 ± 0.012	0.17 ± 0.011	0.166 ± 0.016	0.17 ± 0.013
	Purity	95.7% ± 0.0%	96% ± 0.5%	95.6% ± 0.5%	95.5% ± 0.5%	96% ± 0.7%	**96% ± 0.5%**
	Time required (s)	**5.7e-3 ± 0.7e-3**	1.2 ± 4.66	5.46 ± 15.16	0.19 ± 0.157	1.003 ± 0.225	2.59 ± 0.32
	Dunn Index	1.75 ± 0.00	1.76 ± 0.01	1.76 ± 0.004	**1.78 ± 0.08**	1.734 ± 0.08	1.734 ± 0.007
	Adj. Rand Index	0.834 ± 0.00	0.832 ± 0.012	0.821 ± 0.006	0.76 ± 0.13	0.857 ± 0.015	**0.861 ± 0.006**
	CH Index	**1040 ± 0.00**	1039 ± 0.99	1035 ± 1.55	946 ± 117	1036 ± 3.9	1038 ± 2.11
Diabetes (Correlation)	Classification Entropy	**0.50 ± 0.0**	0.58 ± 0.087	0.54 ± 0.106	0.584 ± 0.058	0.61 ± 0.02	0.59 ± 0.03
	Purity	66.02% ± 0.0%	68.5% ± 1.97%	67.6% ± 5.7%	68.1% ± 2.5%	68.5% ± 2.3%	**71.3% ± 1.5%**
	Time required (s)	**8.1e-3 ± 1.3e-3**	13.15 ± 14.44	11.76 ± 18.43	0.59 ± 0.07	0.44 ± 0.13	2.48 ± 0.48
	Dunn Index	2.5 ± 0.001	2.5 ± 0.003	2.4 ± 0.07	2.5 ± 0.03	2.5 ± 0.009	**2.5 ± 0.9**
	Adj. Rand Index	0.03 ± 0.003	0.04 ± 0.04	0.03 ± 0.04	0.027 ± 0.037	0.029 ± 0.008	**0.0273 ± 0.003**
	CH Index	530.4 ± 5.6	429.3 ± 89	403.6 ± 84.86	389.3 ± 22	526.7 ± 20.9	**533.3 ± 4.8**
Optical Digits (Euclidean)	Classification Entropy	**0.57 ± 0.05**	0.61 ± 0.12	0.68 ± 0.06	0.69 ± 0.06	0.67 ± 0.05	0.61 ± 0.04
	Purity	72.06% ± 8.2%	58.8% ± 17.5%	70.3% ± 4.7%	65.2% ± 9.8%	72.2% ± 6.1%	**75.1% ± 3.7%**
	Time required (s)	**0.39 ± 0.2**	886 ± 229	575 ± 190	17.9 ± 12.1	29.5 ± 8.6	86.7 ± 14.3
	Dunn Index	**0.80 ± 0.12**	0.68 ± 0.26	0.69 ± 0.18	0.73 ± 0.097	0.70 ± 0.07	0.73 ± 0.09
	Adj. Rand Index	0.63 ± 0.06	0.54 ± 0.15	0.59 ± 0.04	0.56 ± 0.06	0.62 ± 0.04	**0.65 ± 0.03**
	CH Index	509.5 ± 17.6	486.1 ± 36.1	468.1 ± 25.8	464.6 ± 28.3	490.7 ± 14.7	**510.2 ± 9.8**
Musk – version 1 & 2 (Euclidean)	Classification Entropy	0.61 ± 0.04	0.63 ± 0.05	0.63 ± 0.0	0.6 ± 0.05	0.6 ± 0.04	**0.59 ± 0.04**
	Purity	58.9% ± 9.3%	56.05% ± 11.3%	53.58% ± 0.0%	62.45% ± 10.5%	63.15% ± 9.9%	**64.78% ± 9.7%**
	Time required (s)	**0.3 ± 0.11**	149.4 ± 3.18	530 ± 128.6	10.75 ± 15.9	32.8 ± 15.4	137.15 ± 29.5
	Dunn Index	1.25 ± 0.13	1.18 ± 0.16	1.17 ± 0.5	1.3 ± 0.152	1.31 ± 0.136	**0.332 ± 0.132**
	Adj. Rand Index	0.017 ± 0.07	0.038 ± 0.05	-0.03 ± 0.00	0.046 ± 0.07	0.048 ± 0.07	**0.061 ± 0.07**
	CH Index	2.6e3 ± 198	2.4e3 ± 305	2.5e3 ± 1.1	2.6e3 ± 251	2.7e3 ± 196	**2.73e3 ± 190**

Reset strategies, RCE^{r+} 2012 produces solutions with levels of purity and repeatability that are, in most cases, substantially greater than those of the other algorithms. It is also seen that performance is further improved with the *Swarm* strategy. In all datasets, *Swarm*{5} RCE^{r+} 2012 achieves the highest purity centroid locations. PSC and mPSC also show relatively good performance for all benchmark datasets, however the time taken by PSC and mPSC to achieve those results was in most cases several orders of magnitude larger. Table 5.6 shows that the new algorithms are significantly faster than the predecessors. For the breast cancer dataset, the time taken per iteration of RCE^{r+} 2012, and *Swarm*{5} RCE^{r+} 2012 compared to PSC are 139.67 and 27.5 times faster, respectively.

TABLE 5.6: Percentage improvement in time taken per iteration relative to PSC

Dataset	**Algorithm** [a]			
	mPSC	RCE 2013	RCE^{r+} 2013	*Swarm*{5} RCE^{r+} 2012
Iris	2.6%	**5978.3%**	5909.8%	813.5%
Glass	3.7%	**2779.4%**	2777.4%	383%
Wine	4.3%	**4963.5%**	4857.2%	752.9%
Breast Cancer	4.78%	**14311.7%**	13967.2%	2750.7%
Diabetes	5.1%	9915.5%	**10012.9%**	1828.3%
Optical Digits	3.5%	**2734.3%**	2723.5%	897.9%
Musk	7.5%	**1332.2%**	1302.9%	380.1%

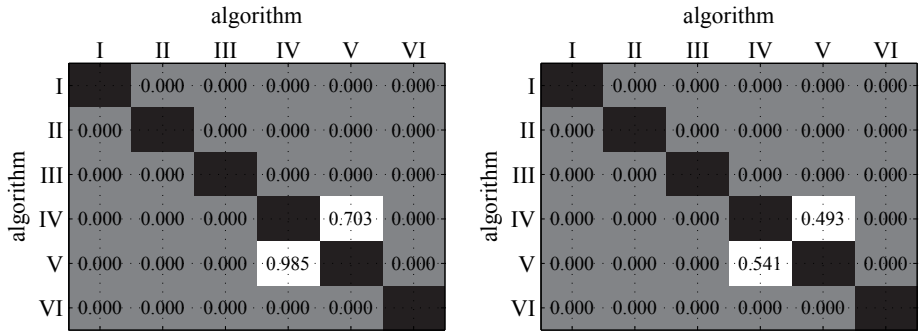
[a] improvement % $= \left(\frac{t_{\text{iter}}^{\text{PSC}}}{t_{\text{iter}}^{\text{algorithm}}} - 1 \right) \times 100\%$

In order to analyze the experimental results regarding the time complexity of each algorithm, a statistical significance test using one-way ANOVA is performed based on the experimental result collected from centroid optimization of the optical digits and musk molecules datasets using the null hypothesis, H_0*: there is no difference between iteration time of algorithm A and algorithm B.* The tests were done using the MATLAB statistics toolbox.

Similarly, the resulting CH cluster validity indices for each algorithm on each dataset were analyzed using the null hypothesis, H_0*: there is no difference between the resulting CH indices of algorithm A and algorithm B.*

In all cases, H_0 is rejected when the p-value is ≤ 0.05. The p-values from the ANOVA tests on the iteration time of optical digits and musk molecules datasets are shown in Figure 5.9. The p-values from the ANOVA tests on the resulting CH cluster validity indices on all benchmark datasets are shown in Figure 5.10. The box-plots are shown in Figure 5.12.

The p-values in Figure 5.9 indicate that the differences in iteration times between all of the pairs of algorithms, except between RCE and RCE^{r+}, are statistically significant.



(A) Optical Digits (B) Musk Molecules

FIGURE 5.9: P-values to three decimal places based on the iteration times given in Table 5.5. Grey boxes indicate $p \leq 0.05$. I. K-means; II. PSC; III. mPSC; IV. RCE 2012; V. RCE^{r+} 2012; VI. *Swarm*{5}RCE^{r+} 2012.

Figure 5.11 shows box-plots comparisons for purity (P) from Table 5.5. Figure 5.11 shows that the RCE^{r+} variants are able to achieve results that are more consistent than the results of K-means, PSC, mPSC, and RCE 2012, with *Swarm*{5} RCE^{r+} 2012 being most consistent.

Figure 5.12 shows box-plot comparisons for CH indices that are given in Table 5.5. Both Table 5.5 and Figure 5.12 show that the RCE^{r+} 2012 variants produce clusters that have CH indices that are, on average, higher than those of K-means, PSC, mPSC, and RCE 2012 on the iris, wine, diabetes, optical digits, and musk molecules datasets. Observing the ANOVA results of the CH indices on the iris, wine, diabetes, and optical digits datasets (Figures 5.10a, 5.10c, 5.10e, and 5.10f), the clusters produced by PSC, mPSC and RCE 2012 are not significantly different from one another. In addition, RCE^{r+} 2012 and *Swarm*{5}RCE^{r+} 2012 produce clusters that are not significantly different from each other for the glass, wine, diabetes and musk molecules datasets (Figures 5.10b, 5.10c, 5.10e, and 5.10g).

Figure 5.13 shows box-plot comparisons for time required for optimization that are given in Table 5.5. In terms of optimization time, Table 5.5 and Figure 5.13 show that k-means is still the computationally lightest algorithm, followed by RCE 2012 and RCE^{r+} 2012. *Swarm*{5} RCE^{r+} 2012 has a longer optimization time than the other RCE variants. PSC and mPSC exhibit the longest optimization times.

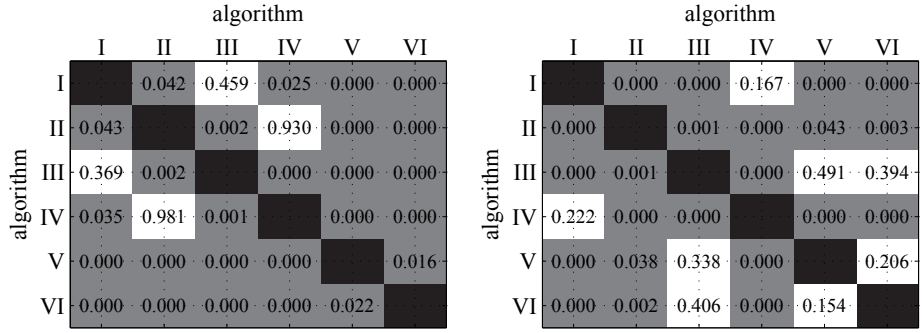
5.8 Conclusion

Rapid Centroid Estimation (RCE) [4] is a light-weight semi-stochastic clustering algorithm which is inspired by Eberhart and Kennedy's Particle Swarm Optimization (PSO) [77] and Cohen and

de Castro Particle Swarm Clustering (PSC) [75]. We introduced the algorithm in 2012 [71, 73] as a lightweight simplification to the PSC algorithm [75]. RCE shares similarly to its deterministic predecessor (k-means algorithm) that it clusters data using the voronoi cells principle. Due to its semi-stochastic nature, the RCE is almost guaranteed to discover the natural local optimum solution given the correct number of clusters, parameters, and sufficient number iterations are supplied.

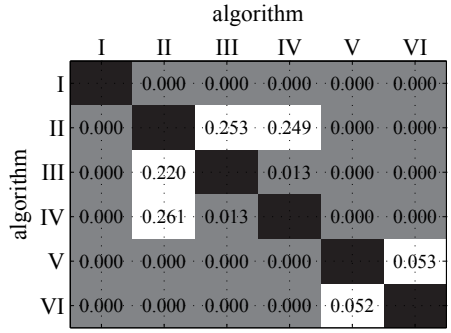
In Definition 5.2.10, we propose a simplified update rule for PSC derived based on the proof in Theorem 4.1. By calculating only the resultant vector, the time complexity for each position update can be reduced up to as low as that of the k-means. Concurrently, this new redefinitions reveals two problems associated with the PSC original formulation: The first problem is concerned with computational complexity of random number generation; The second is concerned with the dilution of randomness as a consequence of Kolmogorov's strong law of large number. This discovery leads to the proposition of Definition 5.2.11 (Simplified Resultant Vector) — the latter is used in RCE — which addresses these issues.

In Theorem 5.1, we prove that the RCE in its fundamental form inherits the consequence of Theorem 4.1 including the curse of initial position. Conceding this fact, we propose strategies including substitution, particle reset, and multi-swarm in order to alleviate the severity of the curse. Especially when the RCE is operated as a multi-swarm, each of the RCE subswarm can be assigned partial random sampled data which greatly reduces computational burden and increases swarm diversity. These characteristics are particularly desirable when the swarm are to be aggregated using consensus/ensemble clustering.

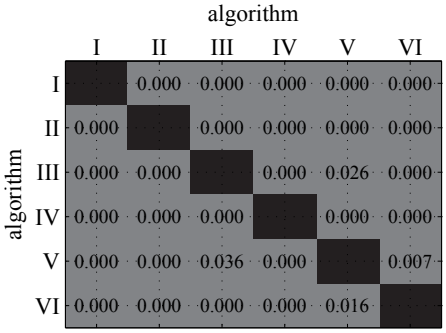


(A) Iris

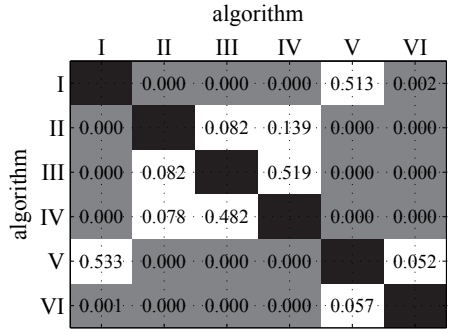
(B) Glass



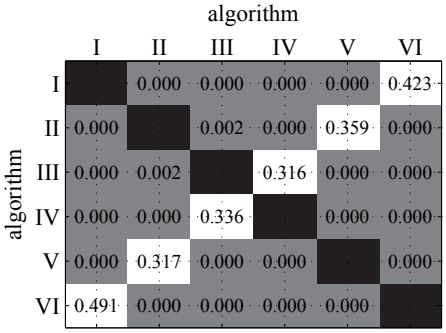
(C) Wine



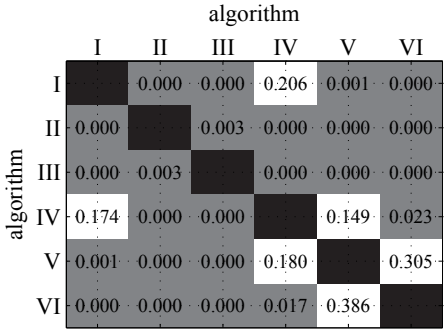
(D) Breast Cancer



(E) Diabetes

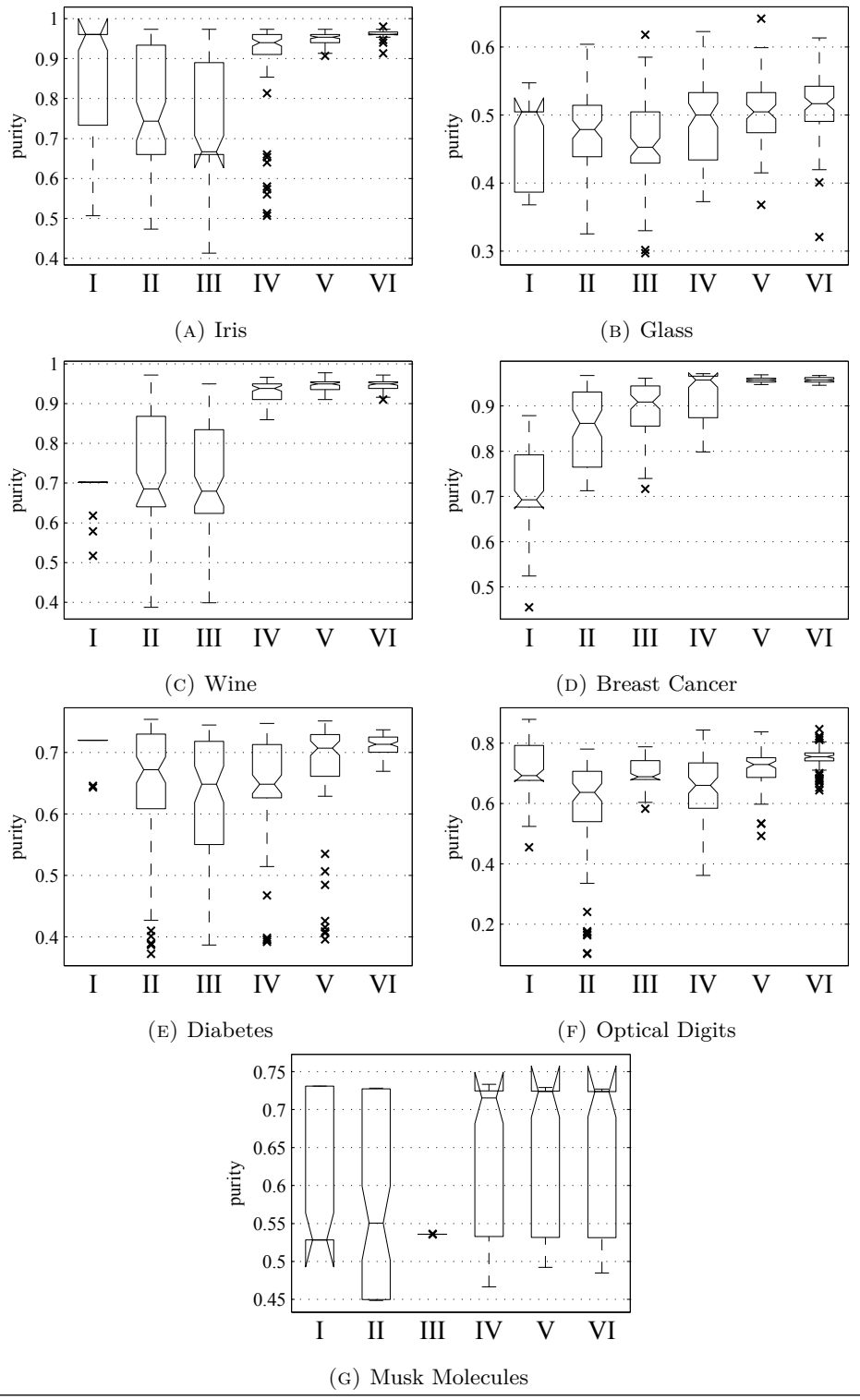


(F) Optical Digits



(G) Musk Molecules

FIGURE 5.10: P-values to three decimal places based on the resulting CH indices on all benchmark datasets given in Table 5.5. Grey boxes indicate $p \leq 0.05$. I. K-means; II. PSC; III. mPSC; IV. RCE 2012; V. RCE^{r+} 2012; VI. *Swarm*{5}RCE^{r+} 2012.



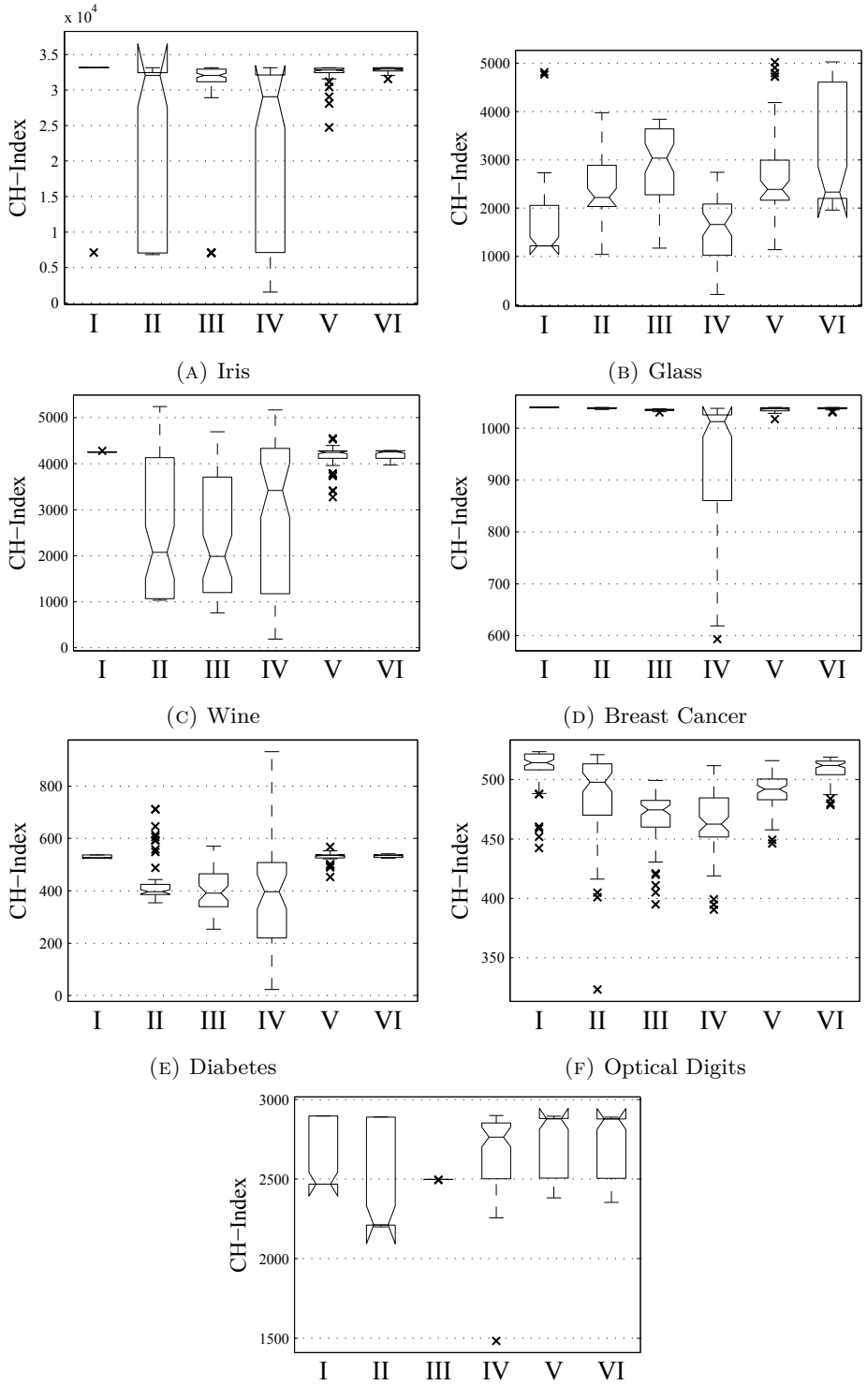
(A) Iris (B) Glass

(C) Wine (D) Breast Cancer

(E) Diabetes (F) Optical Digits

(G) Musk Molecules

FIGURE 5.11: Box-plots [164, 165] of purity values given in Table 5.5. x-axis indicates algorithms: I. K-means; II. PSC; III. mPSC; IV. RCE 2012; V. RCE^{r+} 2012; VI. *Swarm*{5}RCE^{r+} 2012.



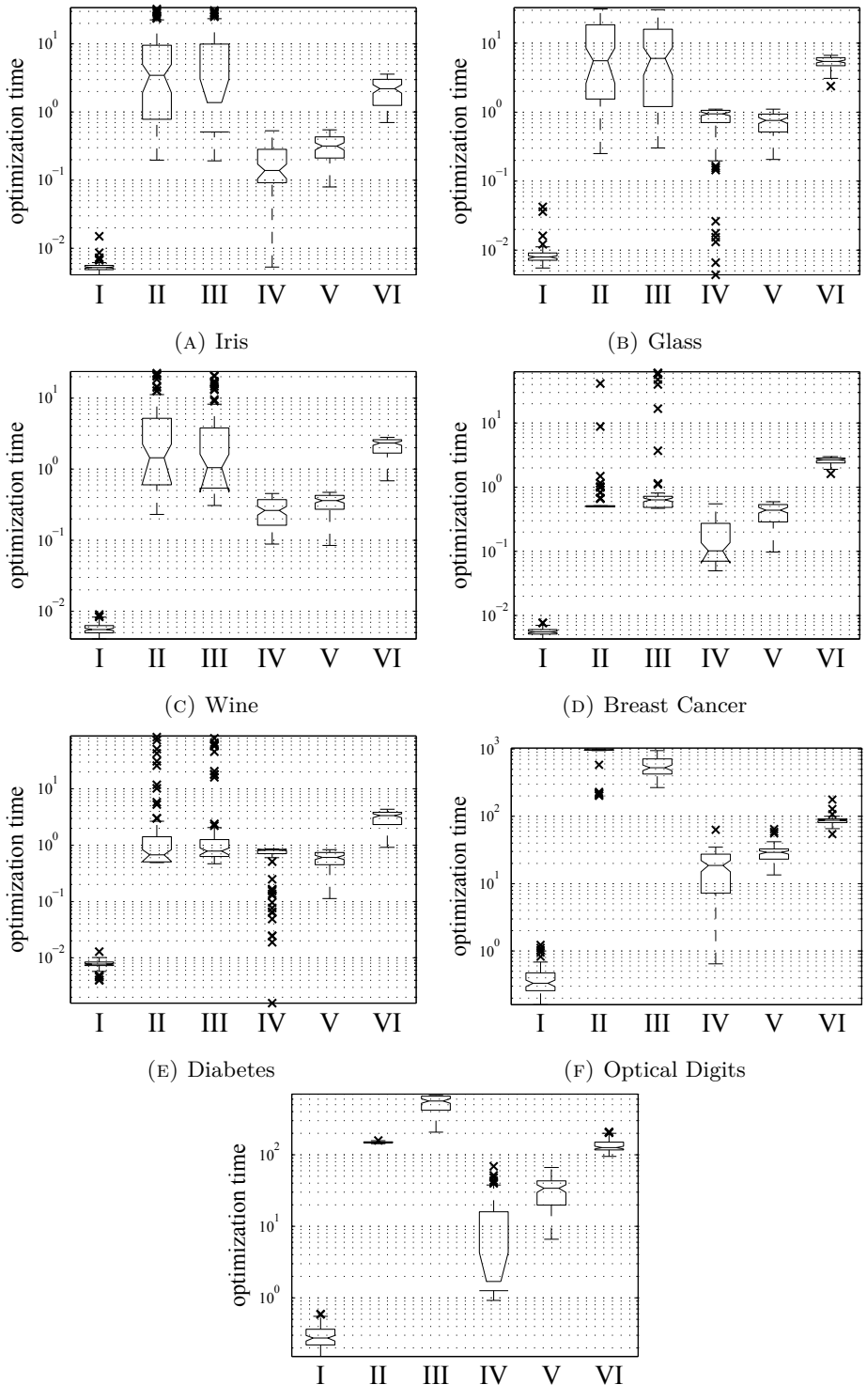
(A) Iris (B) Glass

(C) Wine (D) Breast Cancer

(E) Diabetes (F) Optical Digits

(G) Musk Molecules

FIGURE 5.12: Box-plots [164, 165] of CH indices given in Table 5.5. x-axis indicates algorithms: I. K-means; II. PSC; III. mPSC; IV. RCE 2012; V. RCE^{r+} 2012; VI. *Swarm*{5}RCE^{r+} 2012.



(A) Iris (B) Glass

(C) Wine (D) Breast Cancer

(E) Diabetes (F) Optical Digits

(G) Musk Molecules

FIGURE 5.13: Box-plots [164, 165] of optimization times given in Table 5.5. x-axis indicates algorithms: I. K-means; II. PSC; III. mPSC; IV. RCE 2012; V. RCE^{r+} 2012; VI. *Swarm*{5}RCE^{r+} 2012.

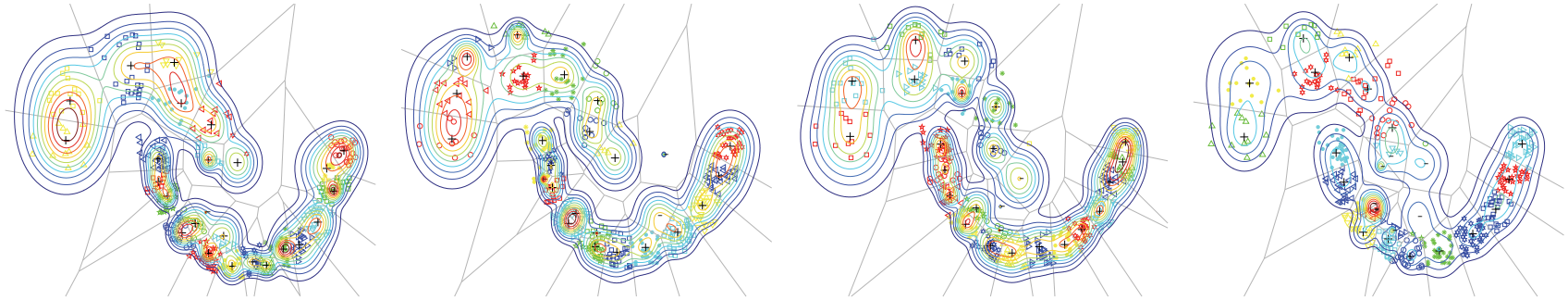
Chapter 6

Ensemble Rapid Centroid Estimation

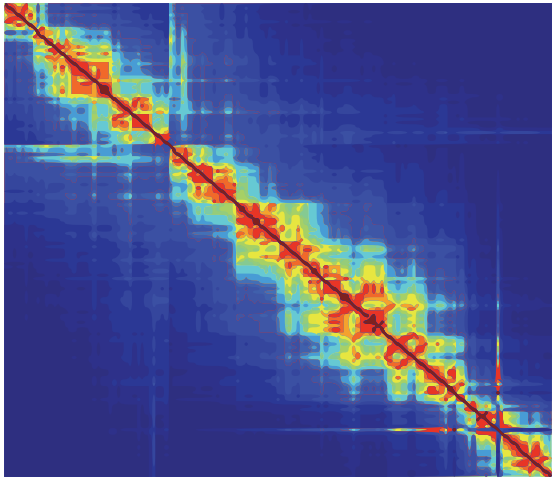
Rapid Centroid Estimation (RCE) has been both theoretically and empirically proven to increase the efficiency of Particle Swarm Optimization (PSO) based clustering algorithms, in particular Cohen - de Castro's Particle Swarm Clustering (PSC) by providing leaner computational complexity and higher stability [4]. However, due to the voronoi tessellation principle, RCE is suitable only for Gaussian clusters. Moreover, the number of clusters also needs to be pre-specified.

The Ensemble RCE (ERCE) sought to address the above limitations. ERCE modifies the concept of swarm in RCE 2014, such that the algorithm can be used efficiently together with conventional ensemble aggregation techniques including Co-Association Tree (CA-tree) [66], Fuzzy Evidence Accumulation [76], and Weighted Evidence Accumulation [65] which allows it to handle non-convex clusters and estimate the number of clusters in larger datasets with quasilinear complexity in both time and space [55]. Interested readers are encouraged to refer to Section 2.6 for further details regarding the three consensus algorithm. The graphical abstract of the algorithm can be seen in Figure 6.1.

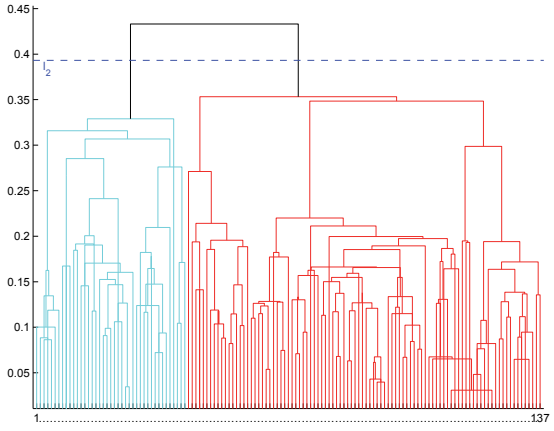
This chapter is organized as follows. Section 6.1 proposes two strategies for improving the diversity of the swarm in an ensemble setting. Section 6.2 provides the guidelines for the application of consensus clustering paradigm to ERCE. Section 6.3 proposes an improvement to ERCE we term the consensus engrams. Section 6.4 provides real world applications of the proposed algorithm on color image segmentation and retinal fundus images. Finally Section 6.5 concludes the chapter.



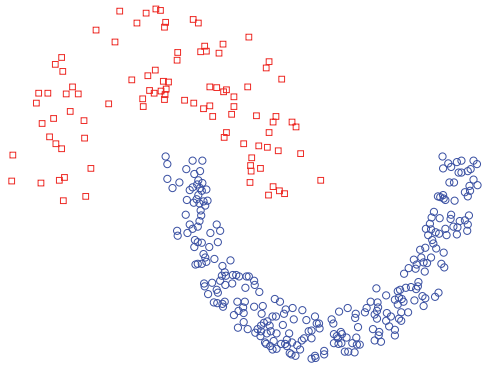
(A) Various fuzzy representations of the data obtained using ERCE. Each swarm in ERCE produces non-identical voronoi tessellations on the data. The degree of fuzziness for each cluster is then optimized using the tradeoff between cluster entropy and the degree of fuzzified dissimilarity [55].



(B) The compressed co-association matrix obtained using the CA-tree [66] and fuzzy WEAC [65, 76] Hybrid [55].



(C) The optimum cut for the corresponding CA-tree using the highest lifetime criterion [64].



(D) Natural grouping is recovered by performing inverse mapping on the CA-tree [55, 66].

FIGURE 6.1: The algorithmic flow of ERCE explained using an artificial dataset [55].

6.1 Improving Swarm Diversity

This section focus mainly on modifying *swarm*$\{n_m\}$ RCE^{r+} to a self-evolving "ensemble swarm". The ERCE equips RCE with the capability to continuously evolve using the concept inspired by Kulis and Jordan's Dirichlet Process (DP) means algorithm [166]. As the result of consensus clustering highly depends on the overall quality of the ensemble partitions [86], proper determination of the quantization parameter k is crucial to ensure appropriate compartmentalization of the data. The following subsections touch upon the ideas on how to elegantly automate such task.

6.1.1 The Concept of "Charged Particles"

This approach is later abandoned due to the relatively high complexity.

Lourenço et al. propose that the presence of redundant partitions in ensemble clustering aggregation may produce an undesireable bias to the final solution [86]. In order to diversify the particles, The concept of *charge* is introduced to create a constant chaotic turbulence in the search space, such that the possibility of creating a duplicate partition is minimized.

There are two types of electric charges – positive and negative. Charges of the opposite polarity will attract one another while charges of the same polarity will repel otherwise. ERCE particles can carry either *positive* or *negative* charge. The initialization is done at random and each particle remains the same charge until the end of the optimization.

Definition 6.1.1 *(Positive Particles).* Positively charged particles are attracted to their member data such that the *self-organizing* resultant vector is positive,

$$\Psi^{+}_{(so)}(t) = +\Psi_{(so)}(t). \tag{6.1}$$

Definition 6.1.2 *(Negative Particles).* Negatively charged particles are repelled by their member data such that the *self-organizing* resultant vector is negative,

$$\Psi^{-}_{(so)}(t) = -\Psi_{(so)}(t). \tag{6.2}$$

Furthermore, negative particles are attracted to their nearby non-empty positive particles,

Although the concept is relatively simple and effective for a number of benchmark datasets [55], there are a number of disadvantages as can be seen in Table 6.1.

TABLE 6.1: Pros and Cons of the Concept of Charged Particles.

Pros	Cons
Charged particles are programmatically easy to implement	The computational and memory complexities are linear to the total number of positive and negative particles,
Negatively charged particles converge to empty spaces which automatically imposes "cannot link" constraint to the yielded ensemble partitions.	The success rate of the method rather depends on the number of positive particles because they are the ones that actively perform the clustering process.
	The charge of each particle is determined at random during initialization, the quality of the final outcome of the ensemble clustering is rather non-deterministic.
	The number of voronoi regions is upper-bounded by K, the total number of particles in a subswarm.

6.1.2 Self-Evolution

Merriam-Webster defines self-evolution as the development by inherent quality or power [43]. The self-evolving swarm RCE equips the swarm with the ability to summon additional particles until the user specified average cluster entropy criterion ξ_Θ is satisfied. The average cluster entropy can be calculated as follows,

$$H_\Theta = \overline{H(\mathbb{X}_\Theta^M, \mathbb{Y}_\Theta)} = \frac{1}{K_\Theta |\mathbb{Y}_\Theta|} \sum_{j \in \mathbb{Y}_\Theta} \sum_{i \in \mathbb{X}_\Theta^M} -u_{ij} \log_2 u_{ij} = \frac{1}{K_\Theta} \mathfrak{V}_H(\mathbb{X}_\Theta^M, \mathbb{Y}_\Theta), \tag{6.3}$$

where u_{ij} denotes the fuzzy membership value of the j^{th} vector in the swarm's self-organizing memory $\mathbb{Y}_\Theta$ with respect to the i^{th} vector in the swarm's local minimum memory $\mathbb{X}_\Theta^M$. $\mathfrak{V}_H$ denotes Bezdek's Cluster Entropy as described in Equation 2.98 in Section 2.4. Implementing self-evolution is relatively straightforward with the following rule,

$$K_\Theta(t) = \begin{cases} K_\Theta(t) + \mathsf{z}_r^+(t) & if \quad \dfrac{1}{K_\Theta(t)} \mathfrak{V}_H(\mathbb{X}_\Theta^M(t), \mathbb{Y}_\Theta) > \xi_\Theta, \\ K_\Theta(t) & otherwise, \end{cases} \tag{6.4}$$

where $K_\Theta(t)$ denotes the number of particles in the swarm Θ at the current iteration t, z_r^+ denotes an upper-bounded random integer, $\mathsf{z}_r^+ \in \mathbb{Z}^+ = [1, 2, \ldots, \mathsf{z}_{max}^+]$, while $\xi_\Theta \in (0, 0.5)$ denotes the prespecified threshold for the average cluster entropy. The desired partition granularity can be specified by varying ξ_Θ as shown in Figure 6.2. The swarm would automatically adjust K until the entropy criterion is satisfied given $\mathbb{Y}_\Theta$ and Θ. When $\xi \to 0$, then theoretically $K_\infty \to K_{\texttt{optimum}}$ for the dataset which is the K that minimizes the average information loss for each cluster [87].

The ensemble partitions using the self-evolution scheme can be seen in Figure 6.3

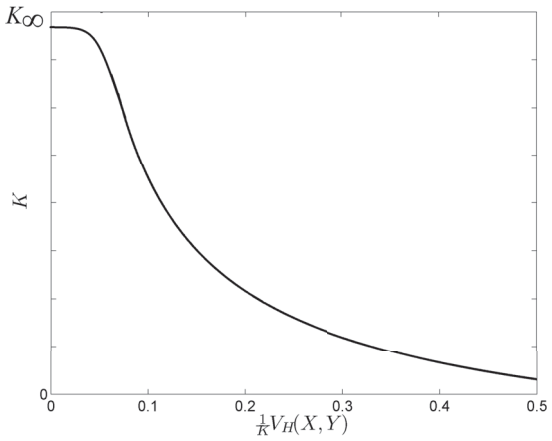
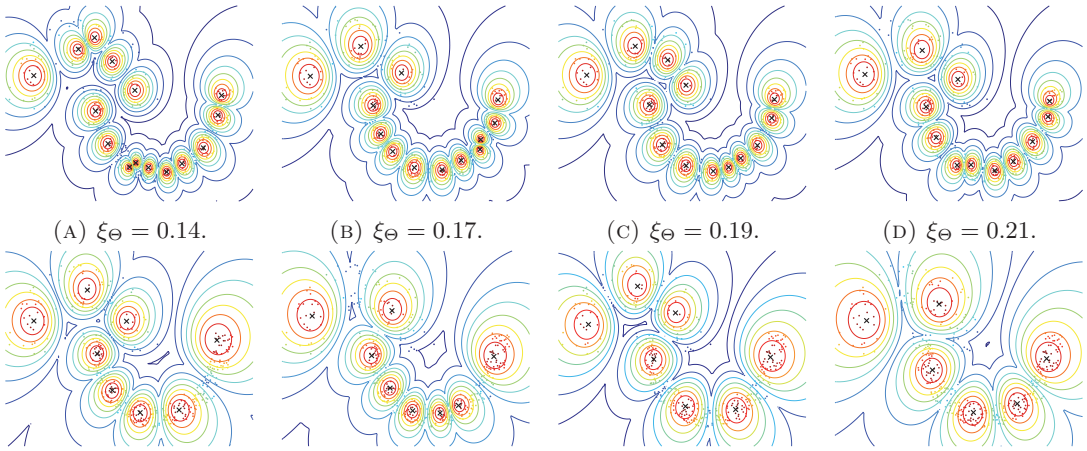


FIGURE 6.2: The typical growth curve of the swarm with varying entropy threshold.



(A) $\xi_\Theta = 0.14$. (B) $\xi_\Theta = 0.17$. (C) $\xi_\Theta = 0.19$. (D) $\xi_\Theta = 0.21$.

(E) $\xi_\Theta = 0.23$. (F) $\xi_\Theta = 0.26$. (G) $\xi_\Theta = 0.28$. (H) $\xi_\Theta = 0.3$.

FIGURE 6.3: Variability of K due to ξ in ERCE swarms on the Half Rings dataset.

The pseudocode for ERCE can be seen in Algorithm 6.24.

6.2 Ensemble Aggregation

The fuzzy ensemble aggregation operates similarly to a typical consensus clustering process. The ERCE ensemble aggregation method uses the CA-tree which allows it to compress redundant label vectors. Using the compression map obtained from the CA-tree, we proceed with compressing the fuzzy membership matrix. The weight for each ERCE partition is quantified using the appropriate cluster validity indices, with higher weight indicates greater quality. The compressed fuzzy weighted

Algorithm 6.24 ERCE

Input: Data points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$, # of swarms n_m, cluster entropy threshold $\xi = \{\xi_1, \ldots, \xi_{nm}\}$, $\lambda_{(so)}, \lambda_{(mi)}$, maximum stagnation δ_{max}, substitution rate ε.
Output: The swarm global optimum $\mathcal{X}^M$ and the corresponding cluster validity $f(\mathcal{X}^M, \mathbb{Y})$.
1: Initialize swarm $\Theta_{1,\ldots,n_m}$.
2: $(\forall \Theta \in \mathbb{S}), \mathbb{Y}_\Theta = randsample(\mathbb{Y}, \eta\%)$.
3: **repeat**
4: **for** $\mathbb{S} = \{\Theta_1, \ldots, \Theta_{n_m}\}$
5: Perform $Swarm\{n_m\}$ RCE^{r+} 2014 update procedures [Algorithm 5.23 lines 5–10],
6: **if** Average cluster entropy $H_\Theta > \xi_\Theta$
7: $K_\Theta(t) \leftarrow K_\Theta(t) + \mathsf{z}_r^+(t)$
8: **end if**
9: **end for**
10: **until** Convergence ($f(\mathcal{X}^M, \mathbb{Y}_\Theta)$ does not improve after n_{max} resets) or maximum iteration reached
11: **return** $\mathcal{X}^M$ and $f(\mathcal{X}^M, \mathbb{Y})$.

consensus matrix $\mathcal{C}_{cfWEAC}$ can be calculated from the compressed fuzzy representation as follows,

$$\mathcal{C}_{cfWEAC} = \frac{\sum_\Theta w_\Theta \mathcal{U}_\Theta^T * \mathcal{U}_\Theta}{\sum_{\Theta=1}^N w_\Theta}, \tag{6.5}$$

where $\mathcal{U}_\Theta$ is the compressed fuzzy membership matrix for the swarm Θ, and $*$ denotes the fuzzy T-norm operator. The consensus partition $\mathsf{U}_\mathcal{C}$ can be obtained from $\mathcal{C}_{cfWEAC}$ and then performing inverse mapping on the CA-tree.

The pseudocode for the ensemble aggregation is listed in Algorithm 6.25. A few results are shown in Figure 6.4. Three of the results presented in this figure are 2 dimensional nonconvex data; The bottom right corner shows a 16 classes 32 dimensional Gaussians. All of which the algorithm managed to recover considerably well.

Algorithm 6.25 Ensemble Aggregation

Input: Fuzzy membership $\mathbf{U}_{\{1,\ldots,n_m\}}$, Crisp membership $\mathsf{U}_{\{1,\ldots,n_m\}}$, Learned covariance matrices Σ.
Output: The consensus partition $\mathsf{U}_\mathcal{C}$.
1: $compression\,map \leftarrow$ CA-tree($\mathsf{U}_{\{1,\ldots,n_m\}}, th$),
2: $\mathcal{U}_{\{1,\ldots,n_m\}} \leftarrow compress(\mathbf{U}_{\{1,\ldots,n_m\}}, compression\,map)$,
3: $w_\Theta \leftarrow \prod \mathfrak{V}(\mathbb{X}_\Theta^M, \mathbb{C}_\Theta)$
4: $\mathcal{C}_\mathcal{U} \leftarrow \dfrac{\sum_\Theta w_\Theta \mathcal{U}_\Theta^T \mathcal{U}_\Theta}{\sum_\Theta w_\Theta}$,
5: $\mathcal{U}_\mathcal{C} \leftarrow GraphPartitioning(\mathcal{C}_\mathcal{U}, \cdots)$
6: **return** $\mathsf{U}_\mathcal{C} \leftarrow decompress(\mathcal{U}_\mathcal{C}, compression\,map)$.

6.2.1 Memory Complexity

To test the memory consumption of various clustering algorithm on a random data (byte precision, volume = 1 Byte – 1 Megabytes, dimension = 2) can be seen in Figure 6.5.

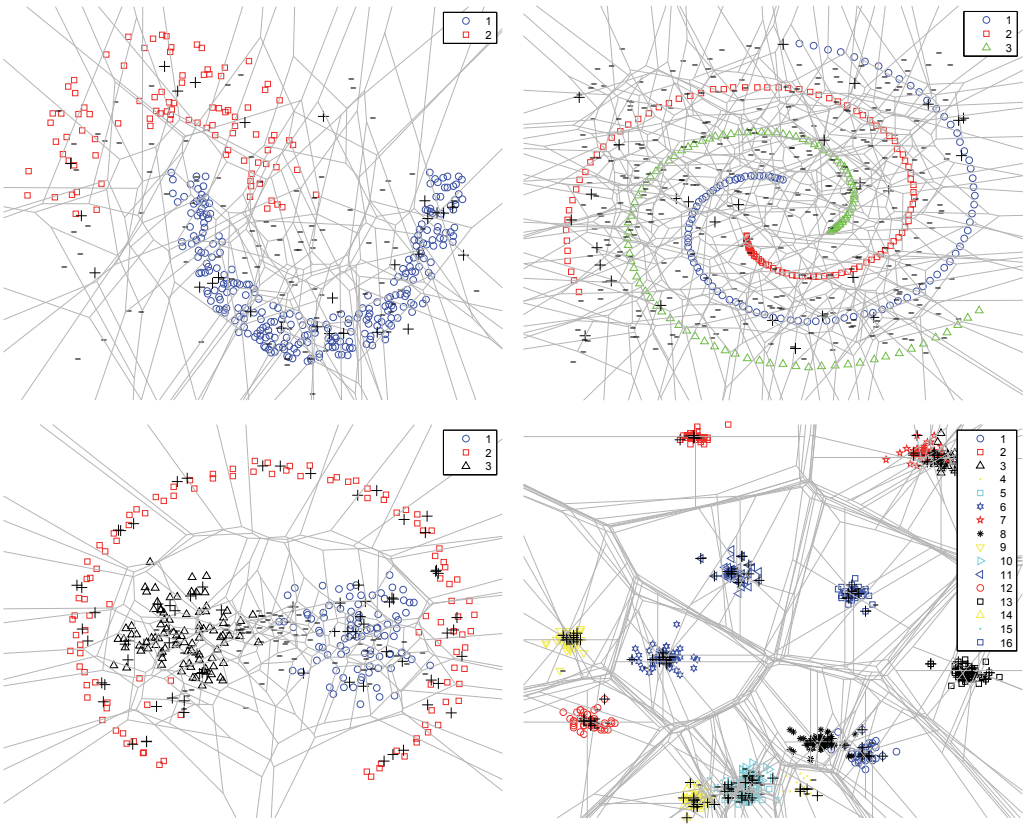


FIGURE 6.4: Clustering Non-convex dataset using ERCE with charged particles

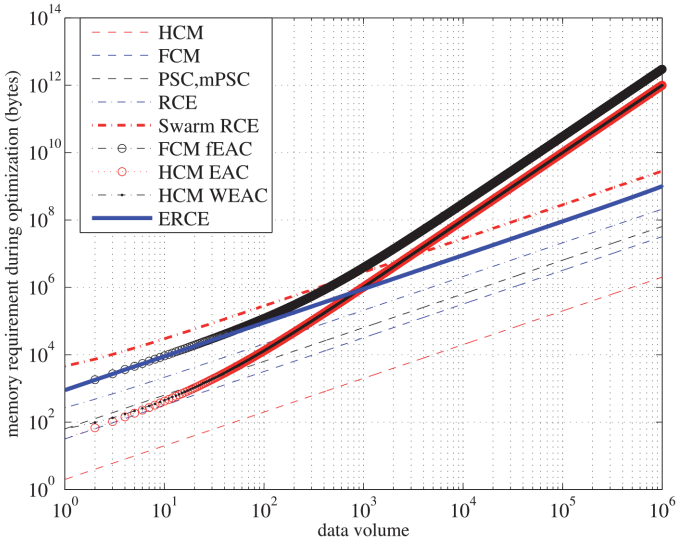


FIGURE 6.5: Memory complexity for clustering 2-dimensional random noise (vol = 1 Byte – 1 Megabytes, d = 2) using various algorithms. The global settings for all algorithms are as follow: the number of representatives (K) = 30; the number of trials/swarms (n_m) = 30.)

It can be seen from Figure 6.5 that the lowest memory requirement for non-ensemble methods is achieved by Hard C-Means (HCM, or k-means), followed by Fuzzy c-Means (FCM), PSC, mPSC, RCE (2012), and Swarm RCE (2012). For ensemble methods, ERCE (Using Swarm RCE 2014) achieves the lowest memory complexity, followed by HCM Evidence Accumulation (EAC), Weighted Evidence Accumulation (WEAC), and FCM with Fuzzy Evidence Accumulation (fEAC). The scalability problem of EAC-based methods are clearly shown in the graph.

6.2.2 Computational Complexity

A known issue with traditional ensemble algorithms — such as the HCM EAC, HCM WEAC, and FCM fEAC, — is that they do not scale well to larger datasets. The main bottleneck for is the EAC algorithm [66].

Realizing this issue, the ERCE make use of the EAC algorithm efficiently on the representative nodes which are extracted using CA-tree. The computational complexity of ERCE will be explained in the following paragraphs.

During the clustering stage, ERCE computational complexity is $\mathcal{O}(n_m KN)$ for a single iteration, where n_m denotes the number of swarms, K denotes the number of particles, and N denotes the number of data. When compared with that of HCM/k-means, it is simply the HCM complexity multiplied by the number of swarms n_m. The complexity for distance fuzzification is $\mathcal{O}(n_m KNL_m)$, where L_m is the number function evaluations required for the m^{th} swarm.

During the ensemble aggregation scheme, the ERCE complexity relies on the complexity of both CA-tree ($\mathcal{O}(N)$[66]) and fuzzy WEAC on the compressed nodes (fcWEAC) $\mathcal{O}(n_m KN_{th}^2)$, where n_m denotes the number of swarms, K denotes the number of particles, and N_{th} denotes the number of representative nodes at a given threshold, $N_{th} < N$. For higher threshold the volume of the node representatives will be much smaller than the dataset, $N_{th} << N$, which in turn makes the fcWEAC assume a quasi-linear complexity, $\mathcal{O}(n_m K log^2(N))$.

The overall complexity of ERCE is therefore,

$$\mathcal{O}(\overbrace{[n_m KN(t_{max} + L_m)]}^{\text{clustering + fuzzification}} + \overbrace{[N]}^{\text{CA-tree}} + \overbrace{[n_m K log^2(N)]}^{\text{fcWEAC}}), \tag{6.6}$$

where t_{max} denotes the predefined number of iterations.

6.3 Consensus Engrams

When dealing datasets that are too large to fit in memory, constructing a complete consensus matrix, even with the help of CA-tree compression, is either very hard if not impossible to do due to the obscene memory requirement. In this scenario the swarm RCE needs to be operated on partial random sampled chunks similarly to the approach used in mini-batch k-means. We propose that it is more efficient to operate on the subswarms which volumes are relatively much more manageable (since $|\mathbb{X}_\theta| = K_\theta$). In order to do so, we serially impose the consensus information from the data and transform it into an affinity value between each consensus nodes.

The concept of node linking shares similarity to the perspective of the Self Organizing Map (SOM) and Hebbian Network [78]. However, contrast to SOM, the internodal linkage in consensus engrams is determined by the "consensus" between subswarms instead of the literal positions of particles in the Euclidean space. This new concept of "closeness" provides a compact representation of clusters as a collection of interconnected nodes, which makes the clustering result robust and much easier to perceive. The efficiency of this approach is demonstrated in practical applications such as data mining and information extraction from images.

6.3.1 Definitions

Definition 6.3.1 *(Consensus Nodes).* Let us define $\mathbb{P}$ as the set of all particles in all subswarms such that

$$\mathbb{P} = \bigcup_{\Theta} \mathbb{X}_\Theta. \tag{6.7}$$

$\mathbb{P}$ can be clustered into C partitions to obtain $\hat{\mathbb{P}} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_C\}$. The consensus nodes are then simply defined in term of expected values of each partition,

$$\mathbb{Z} = \{E[\mathbb{P}|\mathbb{P}_1], \ldots, E[\mathbb{P}|\mathbb{P}_C]\}, \tag{6.8}$$

$$= \{\mathbf{z}_1, \ldots, \mathbf{z}_C\} \in \mathbb{R}^{dim}, \tag{6.9}$$

which is simply the cluster center of each $\mathbb{P}_n$. The consensus nodes can be elegantly obtained using Satuluri and Parthasarathy's Regularized Markov Clustering [153] described in Algorithm 2.10. The affinity between each particles $\mathbf{x} \in \mathbb{P}$ are quantified using the cross-cluster closeness.

Definition 6.3.2 *(Cross-cluster Divergence).* Let $\mathbf{z} \in \mathbb{R}^{dim}$ be defined as a voronoi cell in the voronoi tessellation $\mathbb{Z}$, $\mathbf{x} \in \mathbb{R}^{dim}$ as a voronoi cell in the voronoi tessellation $\mathbb{X}$, and $\mathbf{y} \in \mathbb{R}^{dim}$ as a

data vector randomly sampled from a chunk of data $\mathbb{Y}$. The conditional probabilities,

$$p(\mathbf{y}|\mathbf{z}) = p(\mathbf{z}|\mathbf{y}, \mathbb{Z}), \text{ and} \tag{6.10}$$

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{x}|\mathbf{y}, \mathbb{X}) \tag{6.11}$$

are defined as the responsibility values of the observation $\mathbf{y} \in \mathbb{Y}$ with respect to the voronoi cells $\mathbf{z} \in \mathbb{Z}$ and $\mathbf{x} \in \mathbb{X}$, respectively. The marginal probabilities $p(\mathbf{x})$ and $p(\mathbf{z})$ are simply sum of the conditional probabilities over all $\mathbf{y} \in \mathbb{Y}$ such that $p(\mathbf{z}) = \sum_{\mathbf{y}\in\mathbb{Y}} p(\mathbf{z}|\mathbf{y}, \mathbb{Z}) \leq 1$, and $p(\mathbf{x}) = \sum_{\mathbf{y}\in\mathbb{Y}} p(\mathbf{x}|\mathbf{y}, \mathbb{X}) \leq 1$, by definition. The corresponding empirical probability of $\mathbf{y} \in \mathbb{Y}$ with respect to $\mathbf{z} \in \mathbb{Z}$ and $\mathbf{x} \in \mathbb{X}$ are therefore,

$$\mathcal{P}(\mathbf{y}|\mathbf{z}) = \frac{p(\mathbf{y}|\mathbf{z})}{p(\mathbf{z})} = \frac{p(\mathbf{z}|\mathbf{y}, \mathbb{Z})}{p(\mathbf{z})}, \text{ and} \tag{6.12}$$

$$\mathcal{P}(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{y}, \mathbb{X})}{p(\mathbf{x})}, \tag{6.13}$$

where $\sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{y}|\mathbf{z}) = 1$ and $\sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{y}|\mathbf{x}) = 1$ by definition. The cross-cluster divergence of the voronoi region $\mathbb{Z}$ due to $\mathbb{X}$ given the observations $\mathbb{Y}$ emerges naturally as the KL-divergence between the empirical probabilities,

$$D(\mathbb{Z}, \mathbb{X}|\mathbb{Y}) = KL(\mathcal{P}(\mathbf{y}|\mathbf{z})||\mathcal{P}(\mathbf{y}|\mathbf{x})), \quad \mathbf{y} \in \mathbb{Y} \tag{6.14}$$

$$= \sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{y}|\mathbf{z}) \log \frac{\mathcal{P}(\mathbf{y}|\mathbf{z})}{\mathcal{P}(\mathbf{y}|\mathbf{x})}, \tag{6.15}$$

$$= \sum_{\mathbf{y}\in\mathbb{Y}} \frac{p(\mathbf{y}|\mathbf{z})}{p(\mathbf{z})} \log \frac{p(\mathbf{x})p(\mathbf{y}|\mathbf{z})}{p(\mathbf{z})p(\mathbf{y}|\mathbf{x})} \tag{6.16}$$

Definition 6.3.3 *(Cross-cluster Closeness)***.** The closeness of the approximation of $\mathbb{C}_z$ using $\mathbb{C}_x$ can be calculated by applying a zero mean Gaussian kernel to the cross-cluster divergence,

$$\phi(\mathbf{z}, \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{D(\mathbb{Z}, \mathbb{X}|\mathbb{Y})}{2\sigma^2}\right), \tag{6.17}$$

where σ denotes the standard deviation of the Gaussian kernel, describing the penalty factor. Lower σ provides stricter emphasis on the closeness between partitions which is often more preferable for preserving sensitivity when dealing with non-convex problems.

For larger datasets, observations are given in smaller chunks to decrease the memory and computational complexity. In this situation $\phi(\mathbf{z}, \mathbf{x})$ can be updated iteratively by calculating the

probabilistic union. As observations are mutually exclusive, we can apply addition rule as follows,

$$\phi_{(t+1)} \leftarrow \phi_{(t)} + \phi_{(t-1)} - \phi_{(t)}\phi_{(t-1)}, \tag{6.18}$$

where ϕ_t denotes $\phi(\mathbf{z}, \mathbf{x})$ at the t^{th} iteration.

Definition 6.3.4 *(Consensus Engrams).* In neuropsychology, engrams are means by which memory traces are stored as biophysical or biochemical changes in the brain (and other neural tissues) in response to various external stimuli which leave a lasting trace in the brain [78].

The consensus nodes can be arranged in a fully connected cell assembly, called the consensus engrams. The theory is based on one of Hebb's theories on cell assembly [78] specifically, *"When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell."*

Donald O. Hebb [78].

The responsibility vector of an observation $\mathbf{y}$ due to the voronoi tessellation $\Theta = \bigcup_{\theta} \mathbb{C}_\theta$ consisting of K_Θ voronoi regions can be expressed as a column vector with K_Θ rows,

$$\mathbf{u}_\Theta(\mathbf{y}) = \begin{pmatrix} u_{\mathbf{x}_1} \\ \vdots \\ u_{\mathbf{x}_{K_\Theta}} \end{pmatrix} = \begin{pmatrix} p(\mathbf{x}_1|\mathbf{y}, \Theta) \\ \vdots \\ p(\mathbf{x}_{K_\Theta}|\mathbf{y}, \Theta) \end{pmatrix}, \tag{6.19}$$

where $\sum_{\theta} p(\mathbf{x}_\theta|\mathbf{y}, \Theta) = 1$ by definition. $\mathbf{u}_\mathbb{Z}(\mathbf{y})$ can then be similarly defined in the context of the consensus nodes $\mathbb{Z} = \bigcup_{z} \mathbb{C}_z$,

$$\mathbf{u}_\mathbb{Z}(\mathbf{y}) = \begin{pmatrix} u_{\mathbf{z}_1} \\ \vdots \\ u_{\mathbf{z}_C} \end{pmatrix} = \begin{pmatrix} p(\mathbf{z}_1|\mathbf{y}, \mathbb{Z}) \\ \vdots \\ p(\mathbf{z}_C|\mathbf{y}, \mathbb{Z}) \end{pmatrix}, \tag{6.20}$$

where C denotes the number of consensus nodes.

Let us define the mapping kernel Π_Θ for a swarm Θ as a $C \times K_\Theta$ matrix which linearly maps $\mathbf{u}_\Theta$ to $\mathbf{u}_\mathbb{Z}$,

$$\Pi_\Theta = \begin{pmatrix} \pi(\mathbf{z}_1, \mathbf{x}_1) & \pi(\mathbf{z}_1, \mathbf{x}_2) & \dots & \pi(\mathbf{z}_1, \mathbf{x}_{K_\Theta}) \\ \vdots & \vdots & \ddots & \vdots \\ \pi(\mathbf{z}_C, \mathbf{x}_1) & \pi(\mathbf{z}_C, \mathbf{x}_2) & \dots & \pi(\mathbf{z}_C, \mathbf{x}_{K_\Theta}) \end{pmatrix}, \tag{6.21}$$

where

$$\pi(\mathbf{z}_i, \mathbf{x}_k) = \frac{\phi(\mathbf{z}_i, \mathbf{x}_k)}{\sum_{c=1}^{C} \phi(\mathbf{z}_c, \mathbf{x}_k)}, \tag{6.22}$$

such that each column sums to 1. Note that each subswarm is assigned such kernel in order to map the inputs to the appropriate consensus nodes.

The consensus mapping is the core concept that defines the Consensus Engrams algorithm. The responsibility of the consensus node $\mathbf{z} \in \mathbb{Z}$ in the engrams are determined by how the swarm $\mathbb{S} = \{\Theta_1, \dots, \Theta_{n_m}\}$ maps the information $\mathbf{y} \in \mathbb{Y}$ to the corresponding node $\mathbf{z}$. Each data $\mathbf{y}$ that enters into the engrams has to be routed into these mapping kernels in order to calculate $\mathbf{u}_\mathbb{Z}(\mathbf{y})$ as follows,

$$\mathbf{u}_\mathbb{Z}(\mathbf{y}) = \frac{\overset{C\times K_{\Theta_1}}{\left[\ \Pi_{\Theta_1}\ \right]}\mathbf{u}_{\Theta_1}(\mathbf{y}) + \overset{C\times K_{\Theta_2}}{\left[\ \Pi_{\Theta_2}\ \right]}\mathbf{u}_{\Theta_2}(\mathbf{y}) + \cdots + \overset{C\times K_{\Theta_{n_m}}}{\left[\ \Pi_{\Theta_{n_m}}\ \right]}\mathbf{u}_{\Theta_{n_m}}(\mathbf{y})}{|\mathbb{S}|} \tag{6.23}$$

$$= \frac{1}{|\mathbb{S}|} \sum_{\Theta \in \mathbb{S}} \Pi_\Theta \mathbf{u}_\Theta(\mathbf{y}), \tag{6.24}$$

where $\Pi_{\Theta_1}, \Pi_{\Theta_2}, \dots, \Pi_{\Theta_{n_m}}$ are the mapping kernels previously learned in Equation 6.21. The synaptic response of the consensus nodes due to $\mathbf{y}$ is defined as,

$$\mathbf{R}_\mathbb{Z}(\mathbf{y}) = \mathbf{u}_\mathbb{Z}(\mathbf{y}) * \mathbf{u}_\mathbb{Z}^T(\mathbf{y}), \tag{6.25}$$

where $*$ denotes the fuzzy t-norm operator. The $\mathbf{R}_\mathbb{Z}(\mathbf{y})$ matrix is a $C \times C$ positive definite matrix.

This synaptic response matrix describes the firing pattern of the consensus engrams due to an input vector $\mathbf{y}$. The strength of each response value ranges from $r = \left\{0, \frac{1}{C^2}\right\}$ where higher values indicate stronger link between corresponding node pairs. The consensus engrams are learned by averaging the synaptic responses over as many inputs as possible,

$$\mathbf{A}_\mathbb{Z} = \frac{1}{|\mathbb{Y}|} \sum_{\mathbf{y} \in \mathbb{Y}} \mathbf{R}_\mathbb{Z}(\mathbf{y}) \tag{6.26}$$

The diagonal of the consensus engrams $\mathbf{A}_{\mathbb{Z}}$ are discarded,

$$diag(\mathbf{A}_{\mathbb{Z}}) = 0. \tag{6.27}$$

This matrix can then be interpreted by partitioning it into disjoint subsets using any conventional graph partitioning algorithm. Linkage methods (single/average/ward) or Shi and Malik's normalized cuts [90] are two of many possibilities.

In the special case where the consensus engrams matrix is to be used together with agglomerative clustering, it is usually more convenient to express $\mathbf{A}_{\mathbb{Z}}$ in terms of distance. We recommend the following approach using zero mean Gaussian kernel as follows,

$$\begin{aligned} \mathbf{D}_{\mathbb{Z}} &= diag\left(\sum_{col} \mathbf{A}_{\mathbb{Z}}\right), \text{ (Degree matrix)} \\ \mathbf{A}_{nor} &= \mathbf{D}_{\mathbb{Z}}^{-1/2}\mathbf{A}_{\mathbb{Z}}\mathbf{D}_{\mathbb{Z}}^{-1/2} \\ \mathcal{D}_{\mathbb{Z}} &= \frac{1}{\sqrt{2\pi\,\mathrm{var}(\mathbf{A}_{nor})}} \exp\left(-\frac{\mathbf{A}_{nor}}{2\,\mathrm{var}(\mathbf{A}_{nor})}\right), \end{aligned} \tag{6.28}$$

where $\mathrm{var}(\mathbf{A}_{nor})$ is the variance of all the elements in $\mathbf{A}_{nor}$.

An illustration for comparing the result of the conventional Gaussian neighborhood, ERCE with CA-tree, and ERCE with Consensus Engrams approach on the iris flower dataset can be seen in Figure 6.6. All methods correctly discovered three clusters as can be seen in the matrices.

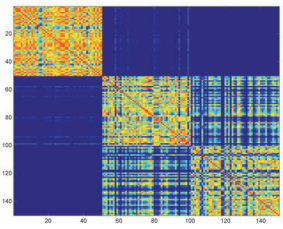
6.3.2 Algorithmic Construct

The overall pseudocode can be summarized in Algorithm 6.26. On line 2 one may realize that symmetrizing the cross-cluster affinity matrix is equivalent to computing the symmetrical KL-divergence across cluster pairs.

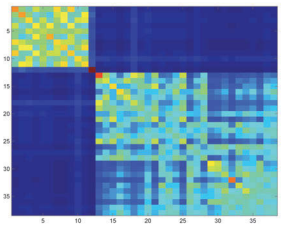
An illustration of the consensus engrams method used on the BIRCH datasets (1 Million points in $\mathbb{R}^2$) [167] are shown in Figure 6.7. Each subswarm were assigned only 15% of the total data. The number of particles for all subswarms were preset at 120.

6.3.3 Complexity Analysis

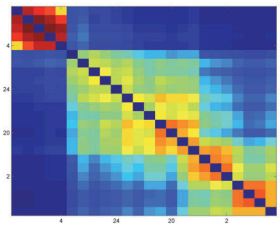
The consensus engram method is designed to run with the bare minimum memory requirement since obviously all the required parameters in the model including $\phi(\mathbb{P}, \mathbb{P})$, mapping kernel Π_{Θ}, and



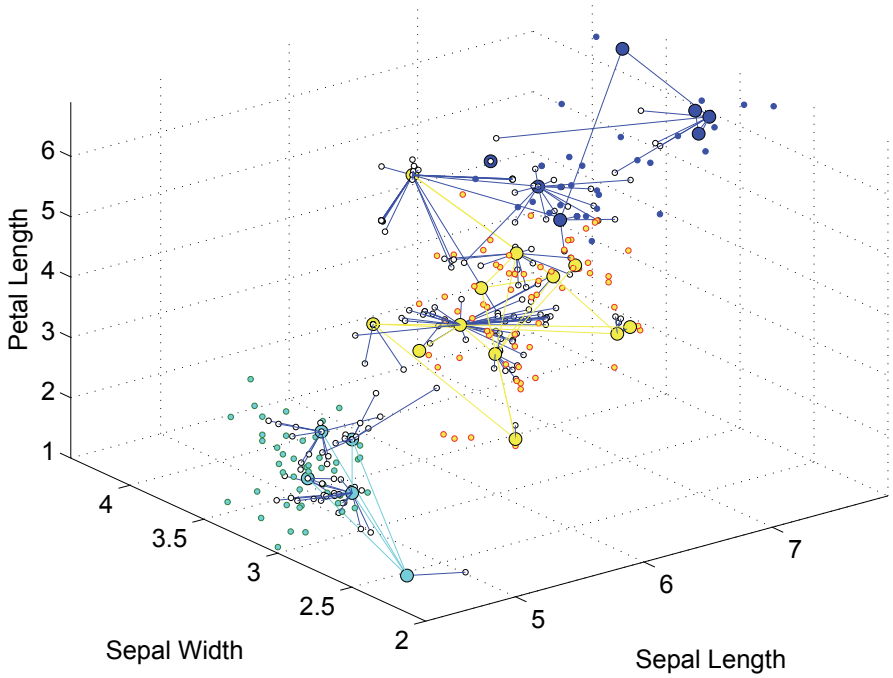
(A) Gaussian Neighborhood.



(B) Compressed Fuzzy Evidence Accumulation matrix (ERCE + CA-tree + fWEAC).



(C) Consensus Engrams matrix $(1 - \mathcal{D}_{\mathbb{Z}})$.



(D) The scatter plot and the corresponding Consensus Engrams.

FIGURE 6.6: Comparison of the graphs produced by Gaussian Neighborhood, ERCE (CA-tree+fWEAC), and ERCE (Consensus Engrams).

consensus node affinity matrix $\mathbf{A}_{\mathbb{Z}}$ depend on neither the volume nor the order of data. In fact, the quality of $\mathbf{A}_{\mathbb{Z}}$ only depends on the variety of data presented, which is directly related to the number of loops. The user task is therefore to select just enough data which may be sufficiently representative.

There are four main components that contributes to the complexity of the consensus engrams algorithm: computation of pairwise cross-cluster closeness $\phi(\mathbb{P}, \mathbb{P})$; construction of consensus nodes (`R-MCL`); calculation of the mapping kernel Π_{Θ}; and computation of $\mathbf{A}_{\mathbb{Z}}$.

From Equation 6.17 it is easy to observe that the computational complexity of cross-cluster closeness between two clusters depends on the volume of a chunk N, the total number of particles in

Algorithm 6.26 Consensus Engrams Construction

Input: Randomly sampled data chunk $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{dim}$

Output: Consensus nodes $\mathbb{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_C\} \in \mathbb{R}^{dim}$, Affinity matrix between consensus nodes $\mathbf{A}_{\mathbb{Z}}$, the consensus mapping kernel Π_{Θ}.

1: $[\Theta_1, \ldots, \Theta_{n_m}; \Sigma_1, \ldots, \Sigma_{n_m}] \leftarrow$ `ERCE`$(\mathbb{Y}_{\eta\%})$

2: $\mathbb{P} \leftarrow \bigcup_{\Theta} \mathbb{X}_{\Theta}$

3: $\mathbf{A}_{\mathbb{P}} \leftarrow \phi(\mathbb{P}, \mathbb{P}) + \phi^T(\mathbb{P}, \mathbb{P})$.

4: $\mathbb{Z} \leftarrow$ `R-MCL`$(\mathbf{A}_{\mathbb{P}})$ [Algorithm 2.10, Section 2.5].

5: Calculate the consensus mapping kernel Π_{Θ} for each subswarm.

6: **for all** $\mathbf{y} \in \mathbb{Y}$

7: $\quad \mathbf{A}_{\mathbb{Z}} \leftarrow \mathbf{A}_{\mathbb{Z}} + \mathbf{R}_{\mathbb{Z}}$

8: **end for**

9: $diag(\mathbf{A}_{\mathbb{Z}}) = 0$ // Discard the diagonal values

10: —— End of Consensus Engrams construction ——

11: The steps below shows a sample partitioning step using Shi-Malik's Normalized Cuts [90]. These steps can be customized according to the use case.

12: $\mathbf{D}_{\mathbb{Z}} = diag\left(\sum_{col} \mathbf{A}_{\mathbb{Z}}\right)$ // Calculate the degree matrix

13: $\mathbf{L}_{normalized} = \mathbf{I} - \mathbf{D}_{\mathbb{Z}}^{-1/2} \mathbf{A}_{\mathbb{Z}} \mathbf{D}_{\mathbb{Z}}^{-1/2}$ // Calculate the graph Laplacian

14: [eigenvectors, eigenvalues] $\leftarrow$ `Lanczos Eigensolver`$(\mathbf{L}_{normalized})$

15: $\mathcal{P} \leftarrow$ `N-cut`$(\mathbf{L}_{normalized})$

16: **return** $\mathbb{Z}$, $\mathbf{A}_{\mathbb{Z}}$, Π_{Θ} and partitioned graph $\mathcal{P}$.

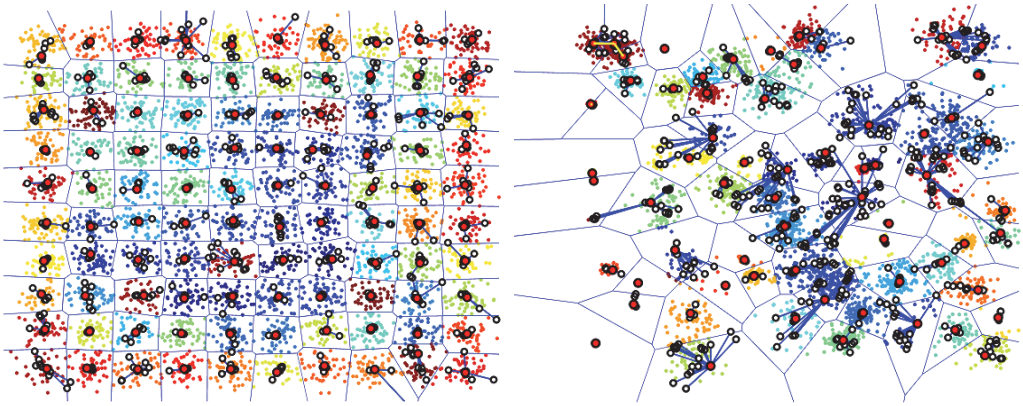


FIGURE 6.7: Finding the natural grouping in the BIRCH [167] datasets using ERCE with consensus engrams. White dots denote particles, while red dots denote consensus nodes. Links were constructed based on cross-cluster divergence.

the swarm $|\forall\Theta| = |\mathbb{P}| = K$. The complexity of calculating $\phi(\mathbb{P}, \mathbb{P})$ is $\mathcal{O}(NK(K-1))$ as a result of the calculation of KL-divergence and exponential kernel between cluster pairs. The complexity for calculating $(\forall\Theta)\Pi_\Theta$ is $\mathcal{O}(NKC)$ due to the calculation of pairwise cross-cluster closeness between the particles $\mathbb{P}$ and the consensus nodes $\mathbb{Z}$. The complexity of $\mathbf{A}_\mathbb{Z}$ arises due to the consensus mapping process $\mathbf{u}_\mathbb{Z}(\mathbf{y}) = \frac{1}{|\mathbb{S}|}\sum_{\Theta\in\mathbb{S}} \Pi_\Theta \mathbf{u}_\Theta(\mathbf{y})$ and the fuzzy t-norm operation $\mathbf{u}_\mathbb{Z}(\mathbf{y}) * \mathbf{u}_\mathbb{Z}^T(\mathbf{y})$. For each chunk $\mathbb{Y}$ the total computational complexity is therefore,

$$\mathcal{O}_{\text{Consensus Engrams}} = \overbrace{NKdim}^{\text{Distance Calculation}} + \overbrace{NK(K-1)}^{\phi(\mathbb{P},\mathbb{P})} + \overbrace{K\kappa^2}^{\texttt{R-MCL}} + \overbrace{NKC}^{\Pi_\Theta \text{ for each } \Theta} + \overbrace{N(KC+C^2)}^{\mathbf{A}_\mathbb{Z}} \quad (6.29)$$

where K denotes the total number of particles in the swarm, C denotes the number of consensus nodes, and κ denotes the average number of non-zero entries in each column of $\mathbf{A}_\mathbb{P}$. The nature of voronoi tessellation guarantees that $\kappa < \frac{K}{|\mathbb{S}|}$, the average number of voronoi regions for each subswarm. The complexity of `R-MCL` is therefore guaranteed to be $< \frac{K^3}{|\mathbb{S}|^2}$.

6.4 Applications

6.4.1 Color Image Segmentation

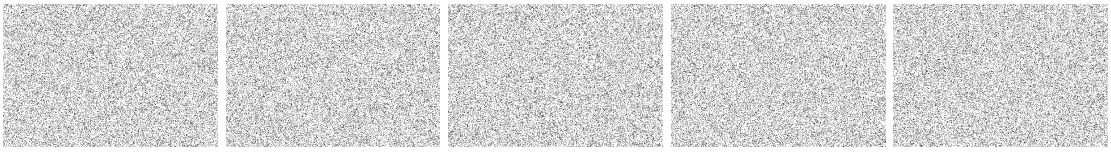
A practical usage of clustering in computer vision is the task of image segmentation and labeling of connected components. Conventional consensus and spectral clustering approaches would require at least $\mathcal{O}(N^2)$ complexity with N denotes the total number of pixels in an image to construct the affinity matrix alone. ERCE + Consensus Engrams allows rapid approximation of connected component from images in linear complexity $\mathcal{O}(N)$.

The images used in this subsection includes various natural segmentation images from Lotus Hill Institute [79] and Berkeley Segmentation Dataset [168]. ERCE was operated using five self-evolving swarms $\xi = 0.03$ – 0.05, each storing only 20% of the total pixels of the image, randomly sampled. The Consensus Engrams were constructed using 20% of the data. An step-by-step illustrative example of the execution of the algorithm are shown in Figure 6.8 and Figure 6.9. It can be observed in Figure 6.9 that the consensus engrams provides an intuitive overview of the statistical property of the features. The experimental results can be seen in Figure 6.10 and Figure 6.11.

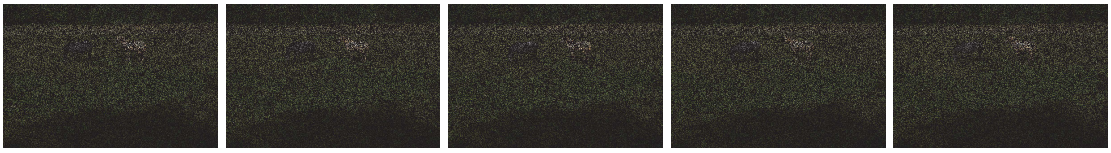
We performed a comparative experiment on images of varying sizes to investigate the time and space complexity of the algorithm. The competing algorithms are strictly consensus/graph-based clustering which includes: ERCE (5 swarms, 20% random sampling, $\xi = 0.03$ – 0.05) with CA-tree

and Fuzzy Weighted Evidence Accumulation (fWEAC); 80 runs of k-means (k is randomized) with Evidence Accumulation (EAC), and; Normalized Cuts with Gaussian Neighborhood. The machine used in this simulation is a laptop computer with Intel Core i5 M520 @ 2.4 Ghz, 4 GB of RAM, running Windows 7. The experimental results are summarized in Figure 6.12.

(A) Original Image.

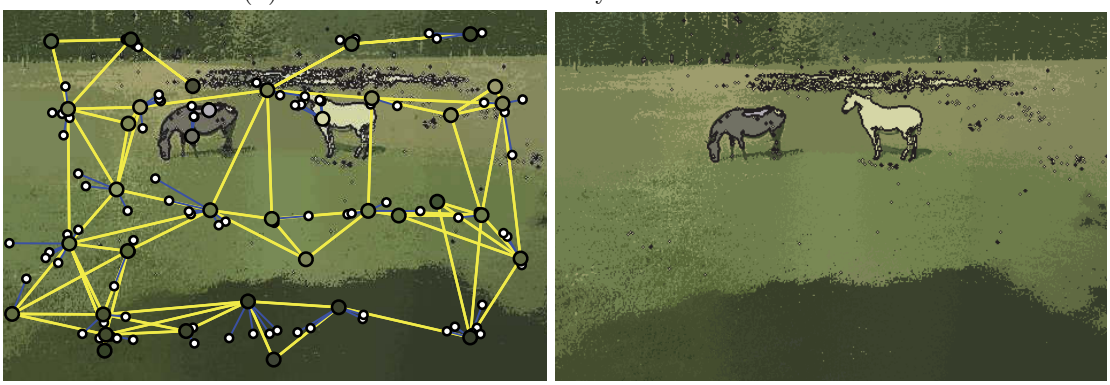


(B) The randomly generated sampling mask for each subswarm.



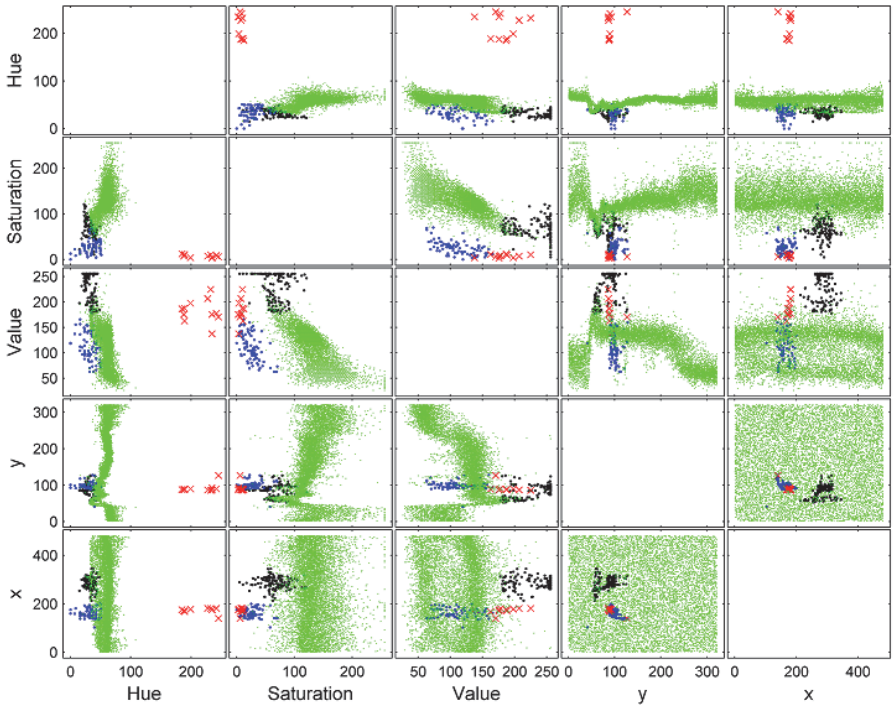
(C) The randomly sampled dataset assigned for each ERCE subswarms, each containing only 20% of $\mathbb{Y}$.

(D) Various Centroids Learned by the ERCE subswarms.

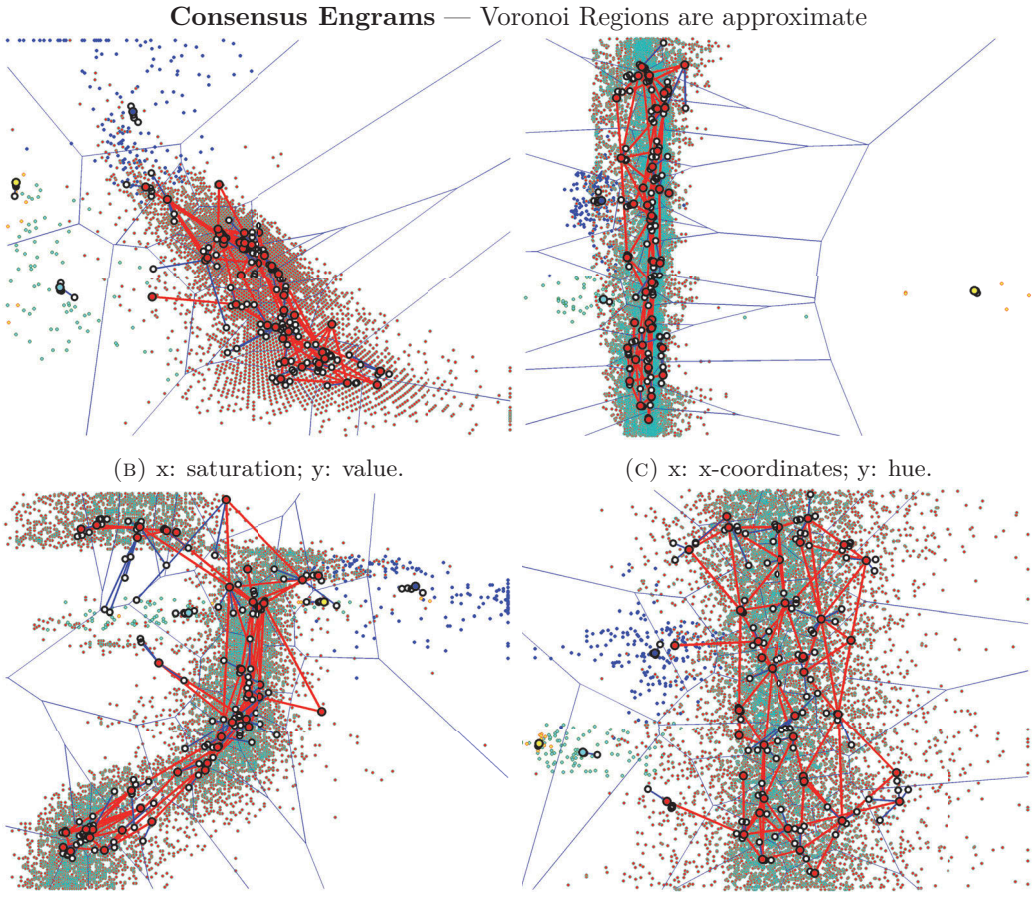


(E) The segmentation result. The Consensus nodes and their linkages are shown on the left image.

FIGURE 6.8: Color Image Segmentation using ERCE (Consensus Engrams Scheme) on an image from Lotus Hill Institute [79]



(A) The scatter plot matrix of the two horses images in Figure 6.8.



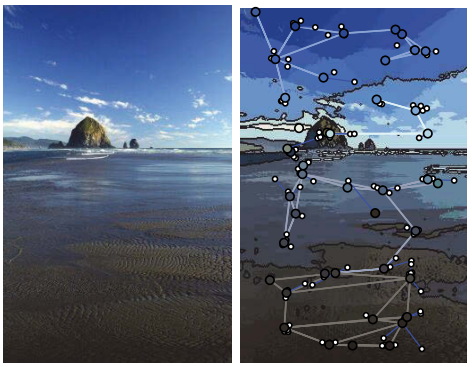
(B) x: saturation; y: value.

(C) x: x-coordinates; y: hue.

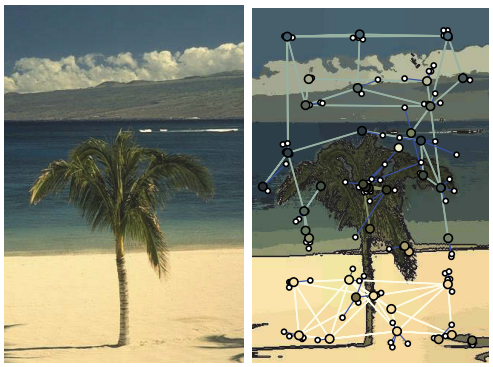
(D) x: value; y: y-coordinates.

(E) x: saturation; y: x-coordinates.

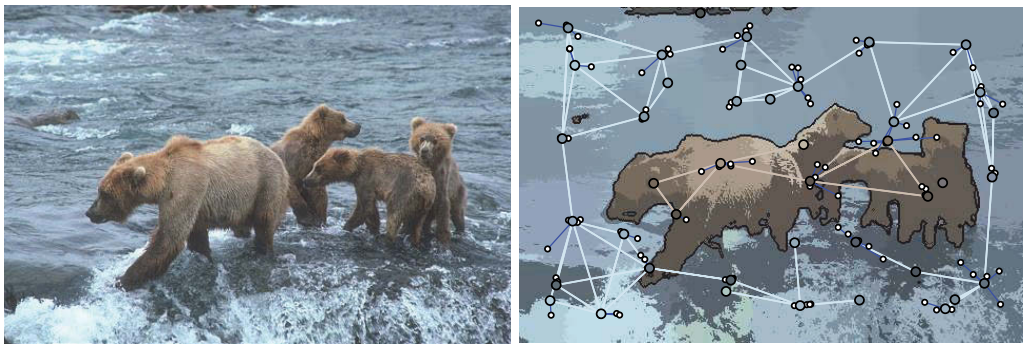
FIGURE 6.9: The scatter plots of the two horses images in Figure 6.8 and its consensus engrams overlays. 4 clusters were discovered.



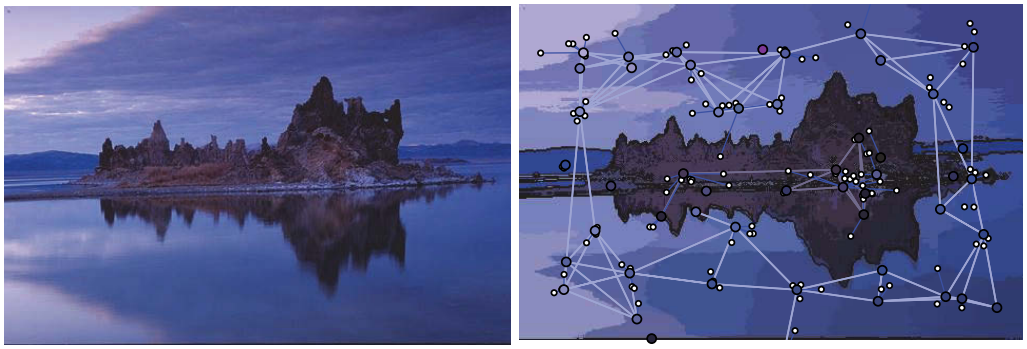
(A) Sea [168].



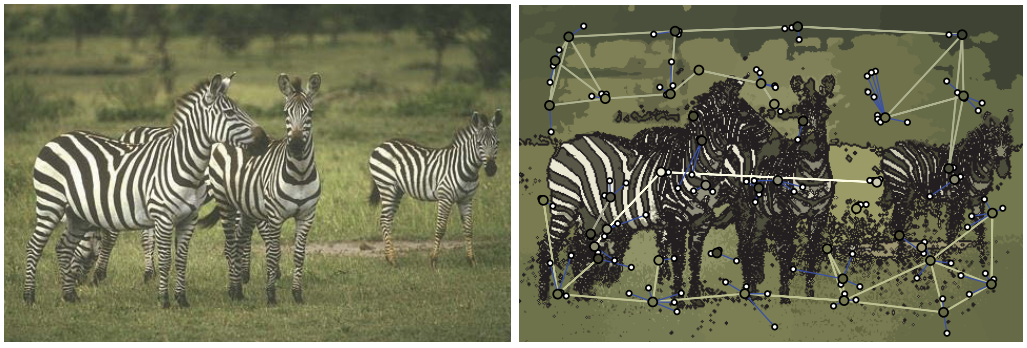
(B) Palm [168].



(C) Bears [168].

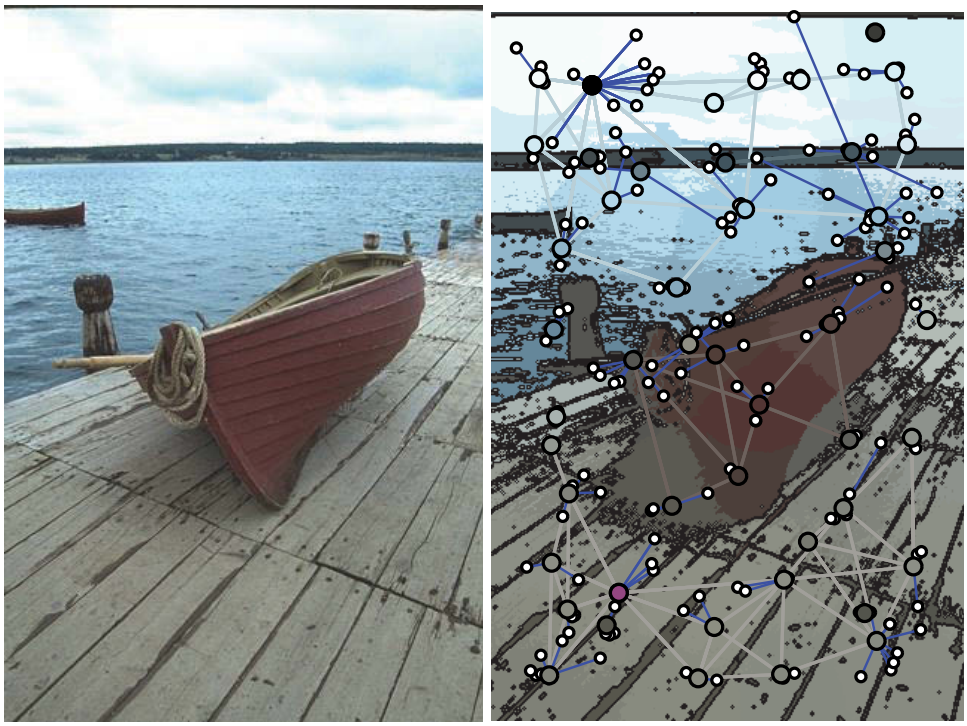


(D) Lake [168].



(E) Zebras [168].

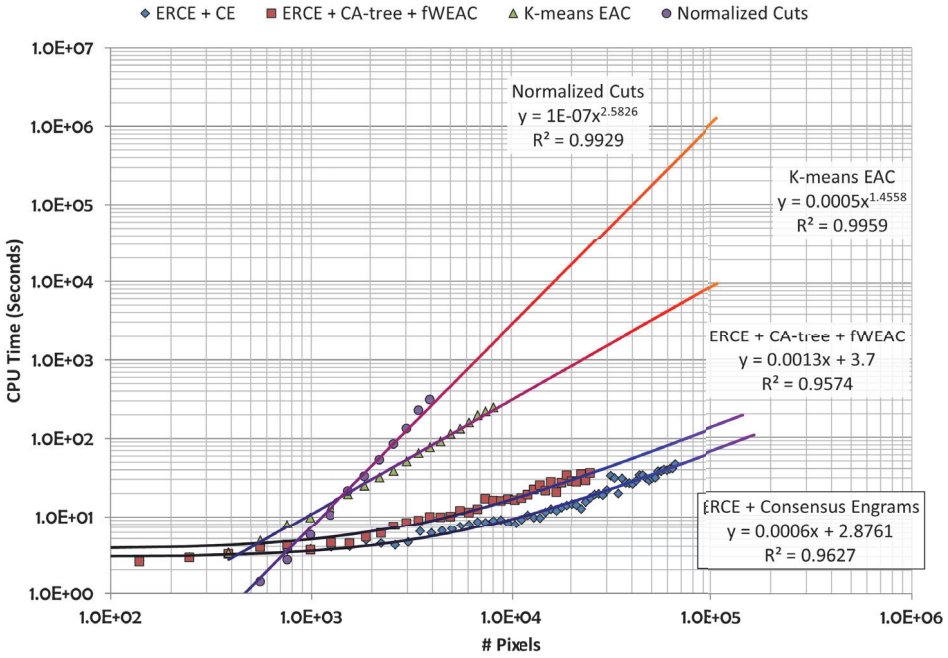
FIGURE 6.10: Image segmentation results using ERCE and their consensus engrams overlay. The graph clustering results are produced using single linkage.



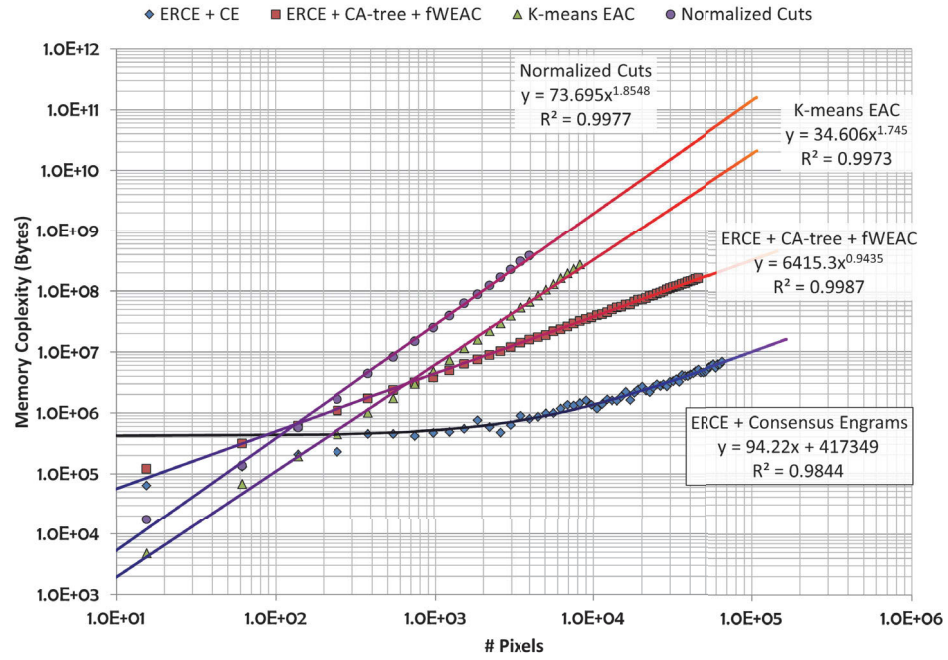
(A) Boat [168].

(B) Temple [168].

FIGURE 6.11: Image segmentation results using ERCE and their consensus engrams overlay — continued. The graph clustering results are produced using single linkage.



(A) Time Complexity.



(B) Memory Complexity.

FIGURE 6.12: The CPU time and memory required of various algorithms with respect to the number of pixels in an image.

The results presented in Figure 6.10 and Figure 6.11 are visually close to the proposed ground truths in [168]. The performance in comparison with graph based algorithms are still unavailable at the present moment since the current machine used for the experiment are incapable of processing the pictures using neither normalized cuts nor k-means EAC. Further investigation will be included in our future works.

From Figure 6.12, we can see that ERCE with Consensus Engrams (CE) were the leanest in terms of both memory and computational complexity. Both the proposed algorithm and ERCE + CA-tree + fWEAC achieved linear complexity in both time and space, with ERCE + CE being significantly leaner. k-means EAC exhibited $\mathcal{O}(N^{1.4558})$ and $\mathcal{O}(N^{1.745})$ computational and memory complexity, respectively. The most expensive algorithm was the normalized cuts which exhibited $\mathcal{O}(N^{2.5826})$ and $\mathcal{O}(N^{1.8548})$ computational and memory complexity, respectively.

6.4.2 Vessel Extraction from Retinal Fundus Images

Another challenge in biomedical informatics is blood vessel extraction from retinal Fundus image. One of the main clinical objectives for retinal vessel segmentation is the early screening of diabetic retinopathy [80]. Diabetic retinopathy is caused by complications of diabetes, which can eventually lead to blindness if not treated. To date, the fundus photography is the only way to detect the early stage of diabetic retinopathy, the non-proliferative diabetic retinopathy (NPDR) [169].

One of the challenges in retinal fundus imagery is to accurately isolate the blood vessels from the other parts of the retina. However, as retinal fundus images are usually complex and noisy, accurate extraction would require the use of advanced technique in computer vision and machine intelligence. We apply our method on the retinal fundus images from the Digital Retinal Images for Vessel Extraction (DRIVE) dataset [80]. The size of each image is 584×565 pixels.

Blood vessels are generally more pronounced in the green channel compared to the other color channels [53]. We perform a morphological closing with a 15×15 disc structuring element on the channel, which is large enough to blur the blood vessels and obtaining the overall background of the channel. This background is then subtracted out of the green channel such that the foreground (blood vessels) are isolated in bright pixels. In addition to the inverted green channel, morphologic features for each pixel in the foreground image were extracted [53, 57]. The first feature was ridge features obtained using Zana's geodesic approach [53]; The second was line features using multiple convolutions of Nguyen's oriented multiscale line filters with $L = \{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$ [57]. The visualization of the features and segmentation result are presented in Figure 6.14.

Using these three features, pixels corresponding to blood vessels can be distinguished using ERCE + CE. The ERCE swarm is set to self-evolving mode with target cluster entropy $\xi = \{0.1 \text{ - } 0.3\}$. The swarm size is set to 5 subswarms, each memorizes 10% of the data, randomly sampled. The consensus engrams is also constructed using only 10% of the data. The chosen metric is the squared Euclidean distance. The conversion of distance to membership uses Bezdek's fuzzification method with the fuzzifier parameter $m = 2$. The consensus engrams σ parameter for the cross-cluster closeness calculation is set to 0.3. Shi-Malik's normalized cuts [90] is chosen as the graph clustering method, forcing 2 clusters using 2-way cut. The final fuzzy value (u_1, u_2) for each pixel $\mathbf{y}$ with respect to the first and second consensus nodes ($\mathbf{z}_1$ and $\mathbf{z}_2$) are then calculated using Equation 6.24 such that,

$$\begin{pmatrix} u_1(\mathbf{y}) \\ u_2(\mathbf{y}) \end{pmatrix} = \frac{1}{|\mathbb{S}|} \sum_{\Theta \in \mathbb{S}} \begin{pmatrix} \pi(\mathbf{z}_1, \mathbf{x}_1) & \pi(\mathbf{z}_1, \mathbf{x}_2) & \dots & \pi(\mathbf{z}_1, \mathbf{x}_{K_\Theta}) \\ \pi(\mathbf{z}_2, \mathbf{x}_1) & \pi(\mathbf{z}_2, \mathbf{x}_2) & \dots & \pi(\mathbf{z}_2, \mathbf{x}_{K_\Theta}) \end{pmatrix} \begin{pmatrix} u_{\mathbf{x}_1}(\mathbf{y}) \\ \vdots \\ u_{\mathbf{x}_{K_\Theta}}(\mathbf{y}) \end{pmatrix}, \tag{6.30}$$

$$\pi(\mathbf{z}_i, \mathbf{x}_k) = \frac{\phi(\mathbf{z}_i, \mathbf{x}_k)}{\sum_{c=1}^{C} \phi(\mathbf{z}_c, \mathbf{x}_k)}, \tag{6.31}$$

where $\mathbb{S}$ denotes the ERCE swarm $\mathbb{S} = \{\Theta_1, \dots, \Theta_{n_m}\}$ and $\pi(\mathbf{z}_i, \mathbf{x}_k)$ denotes the column normalized cross-cluster closeness $\phi(\mathbf{z}_i, \mathbf{x}_k)$ of $\mathbf{z}_i$ approximated by $\mathbf{x}_k$. $u_{\mathbf{x}_i}$ denotes the cluster membership of the i^{th} voronoi region in the subswarm Θ, where $(\forall j), \sum_i u_{\mathbf{x}_i, j} = 1$ by definition. u_1 and u_2 denotes the fuzzy membership of the first (vessels) and second (background) cluster, respectively, where $(\forall j), u_{1j} + u_{2j} = 1$.

The cluster corresponding to the blood vessels are naturally identified as the cluster with the least size $I_{\text{vessels}} = \arg\min_i |\mathbb{C}_i|$ where $|\mathbb{C}_i| = \sum_j u_{ij}$, i denotes the cluster index, j denotes the pixel index.

We propose the optimal threshold value for the algorithm is the ratio between the size of the two clusters, $|\mathbb{C}_{\text{vessels}}|$ and $|\mathbb{C}_{\text{background}}|$,

$$th = \frac{|\mathbb{C}_{\text{vessels}}|}{|\mathbb{C}_{\text{background}}|} = \frac{\sum_j u_{\text{vessels},j}}{\sum_j u_{\text{background},j}}, \tag{6.32}$$

where j denotes the pixel index. This formula is derived as follows. Suppose the two clusters $\mathbb{C} = \{V, B\}$ foreground (blood vessels) and background, respectively. The probabilities are therefore,

$$p(V) = \sum_{\mathbf{y} \in \mathbb{Y}} p(\mathbf{y}|V), \tag{6.33}$$

$$p(B) = \sum_{\mathbf{y} \in \mathbb{Y}} p(\mathbf{y}|B), \tag{6.34}$$

$$p(V) < p(B), \tag{6.35}$$

$$p(V, B) = p(B) - p(V). \tag{6.36}$$

The degree of contamination is simply,

$$\begin{aligned} \% \text{ Contamination} &= \frac{p(B) - p(V, B)}{p(B)} \\ &= \frac{p(V)}{p(B)}, \end{aligned} \tag{6.37}$$

which reminiscent to the signal to noise ratio (SNR) calculation. A descriptive illustration can be seen in Figure 6.13.

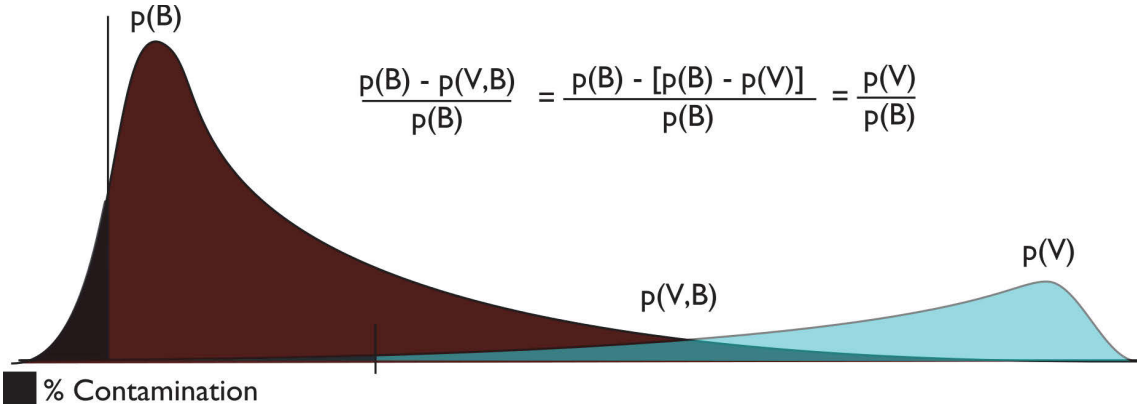


FIGURE 6.13: Illustration of % contamination (not to scale).

The scatter plots showing the features and the consensus engrams overlay are shown in Figure 6.15.

The machine used in this simulation is a laptop computer with Intel Core i5 M520 @ 2.4 Ghz, 4 GB of RAM, running Windows 7. The original data from the DRIVE datasets [80] are presented in Figure 6.16 and Figure 6.17, while the corresponding results are shown in Figure 6.18 and Figure 6.19.

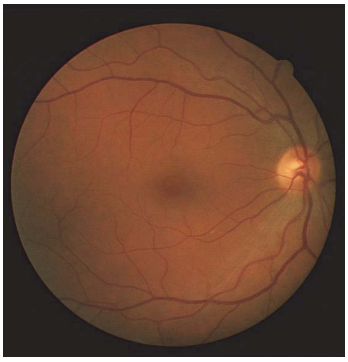
The DRIVE database provides two manually traced gold standard annotations referred to as observer 1 and observer 2. We measure the sensitivity, specificity and accuracy of the proposed

method on each image against the provided observer 1. The average area under the ROC curve against observer 1 is also reported. The local accuracy (where only the vessels and background pixels around the true vessels are considered for accuracy measurement) is also measured using morphological dilation operator with a structuring element of size 3 as proposed by [57]. The correctness of the segmentation were analyzed using Cohen's Kappa statistics. The kappa value measures the degree of agreement between observer 1 and observer 2 given each segmentation result. The overall statistics of the results are then appended using the reported values in [57] and the DRIVE website. The individual result for each image summarized after 20 extraction trials is presented in Table 6.2. The comparative summary against other methods is summarized in Table 6.3. The resulting ROC graph compared to the other approaches are shown in Figure 6.20.

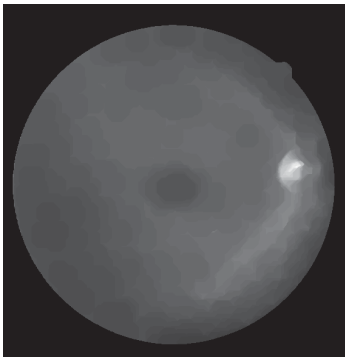
TABLE 6.2: Fundus Image Clustering Results Against the Gold Standard (Observer 1) summarized after 20 trials.

#	Sensitivity	Specificity	Accuracy	#	Sensitivity	Specificity	Accuracy
1	75.3% ± 0.6%	97.8% ± 0.2%	95.8% ± 0.0%	21	75.1% ± 0.1%	98.2% ± 0.0%	96.5% ± 0.0%
2	77.7% ± 0.2%	98.2% ± 0.0%	96.1% ± 0.0%	22	75.1% ± 0.1%	97.7% ± 0.0%	95.7% ± 0.0%
3	75.3% ± 0.5%	97.1% ± 0.2%	95.0% ± 0.1%	23	78.3% ± 0.8%	92.9% ± 0.5%	92.4% ± 0.3%
4	75.2% ± 0.2%	98.1% ± 0.0%	96.1% ± 0.0%	24	75.2% ± 0.2%	97.5% ± 0.0%	95.0% ± 0.0%
5	75.2% ± 0.2%	97.8% ± 0.0%	95.8% ± 0.0%	25	75.2% ± 0.3%	96.3% ± 0.1%	94.4% ± 0.1%
6	75.1% ± 0.1%	97.2% ± 0.2%	95.1% ± 0.0%	26	75.1% ± 0.2%	96.9% ± 0.1%	95.1% ± 0.0%
7	75.2% ± 0.1%	97.1% ± 0.0%	95.2% ± 0.0%	27	75.2% ± 0.2%	97.7% ± 0.0%	95.7% ± 0.0%
8	75.3% ± 0.3%	97.0% ± 0.1%	95.2% ± 0.0%	28	75.2% ± 0.1%	97.6% ± 0.0%	95.4% ± 0.0%
9	75.1% ± 0.2%	97.8% ± 0.0%	96.0% ± 0.0%	29	75.1% ± 0.2%	97.7% ± 0.0%	95.9% ± 0.0%
10	75.1% ± 0.1%	97.7% ± 0.0%	95.9% ± 0.0%	30	75.2% ± 0.2%	97.1% ± 0.0%	95.4% ± 0.0%
11	75.1% ± 0.2%	96.9% ± 0.1%	95.1% ± 0.0%	31	75.4% ± 0.6%	97.2% ± 0.3%	96.0% ± 0.0%
12	75.3% ± 0.2%	97.5% ± 0.0%	95.6% ± 0.0%	32	75.1% ± 0.2%	98.1% ± 0.0%	96.3% ± 0.0%
13	75.1% ± 0.2%	97.2% ± 0.0%	95.1% ± 0.0%	33	75.2% ± 0.2%	98.1% ± 0.0%	96.3% ± 0.0%
14	75.5% ± 0.9%	97.8% ± 0.2%	96.1% ± 0.1%	34	75.2% ± 0.1%	90.3% ± 0.1%	89.0% ± 0.0%
15	75.2% ± 0.1%	97.5% ± 0.0%	96.0% ± 0.0%	35	75.2% ± 0.2%	97.7% ± 0.0%	95.7% ± 0.0%
16	75.2% ± 0.2%	97.2% ± 0.0%	95.3% ± 0.0%	36	75.2% ± 0.2%	97.0% ± 0.0%	94.7% ± 0.0%
17	75.2% ± 0.9%	96.7% ± 0.8%	95.0% ± 0.2%	37	75.1% ± 0.2%	96.7% ± 0.0%	94.9% ± 0.0%
18	75.2% ± 0.1%	96.9% ± 0.0%	95.3% ± 0.0%	38	75.2% ± 0.3%	97.0% ± 0.0%	95.1% ± 0.0%
19	76.7% ± 3.5%	98.0% ± 0.7%	96.5% ± 0.3%	39	75.1% ± 0.3%	97.0% ± 0.0%	95.2% ± 0.0%
20	75.1% ± 0.2%	97.5% ± 0.0%	95.9% ± 0.0%	40	75.2% ± 0.4%	97.8% ± 0.1%	96.1% ± 0.0%

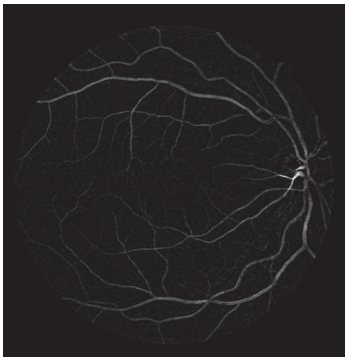
The statistical summary in Table 6.3 shows that the proposed method is superior in all performance aspects compared to the other methods in the literatures. Using both Zana's and Nguyen's feature together with the inverted green channel, ERCE + CE consensus clustering algorithm outperforms even Staal's supervised approach to vessel segmentation (proposed: 95.58% ± 0.51%; Staal: 94.42% ± 0.65%). We attribute this to the highly accurate feature extraction methods by Zana [53] and Nguyen [57].



(A) Fundus Image.



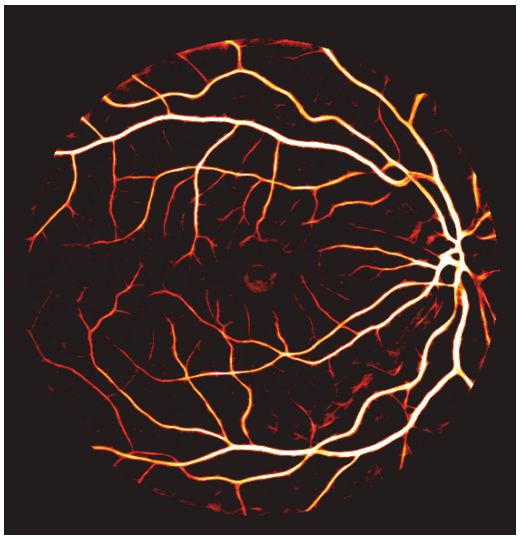
(B) Smoothed Green Channel (Background).



(C) Foreground = Green Channel - Background.

(D) Inverted Green Channel

(E) Zana's Geodesic Morphology and Curvature filter [53].



(F) Summation of multiple convolutions of Nguyen's oriented multiscale line filters $L = \{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$ [57].

(G) Classification Result using ERCE + CE. Sensitivity = 82.5%; Specificity = 96.64%; Accuracy = 95.56%

FIGURE 6.14: Feature extraction from a retinal fundus image.

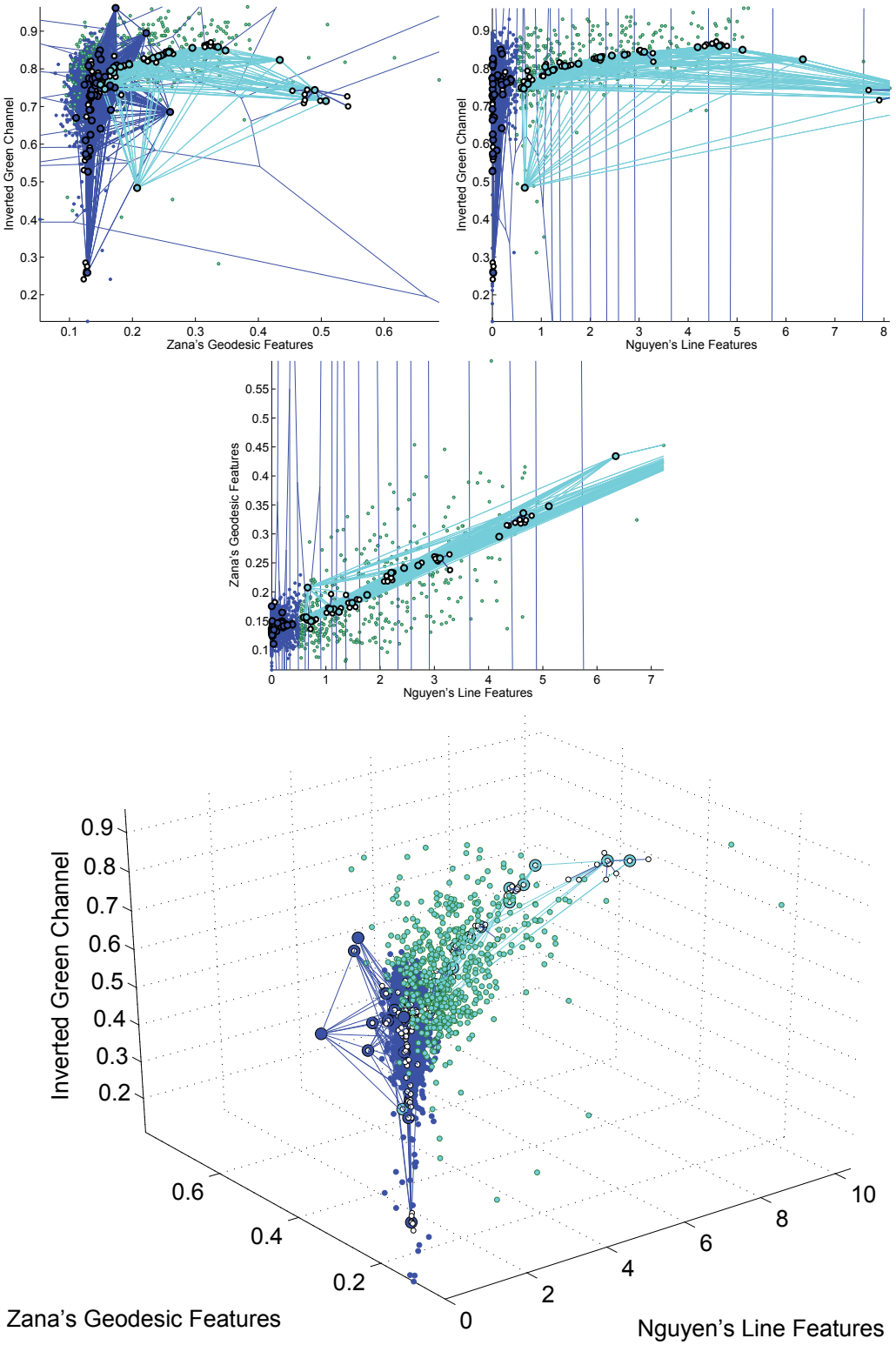


FIGURE 6.15: Scatter plot of the extracted features with the Consensus Engrams overlay.

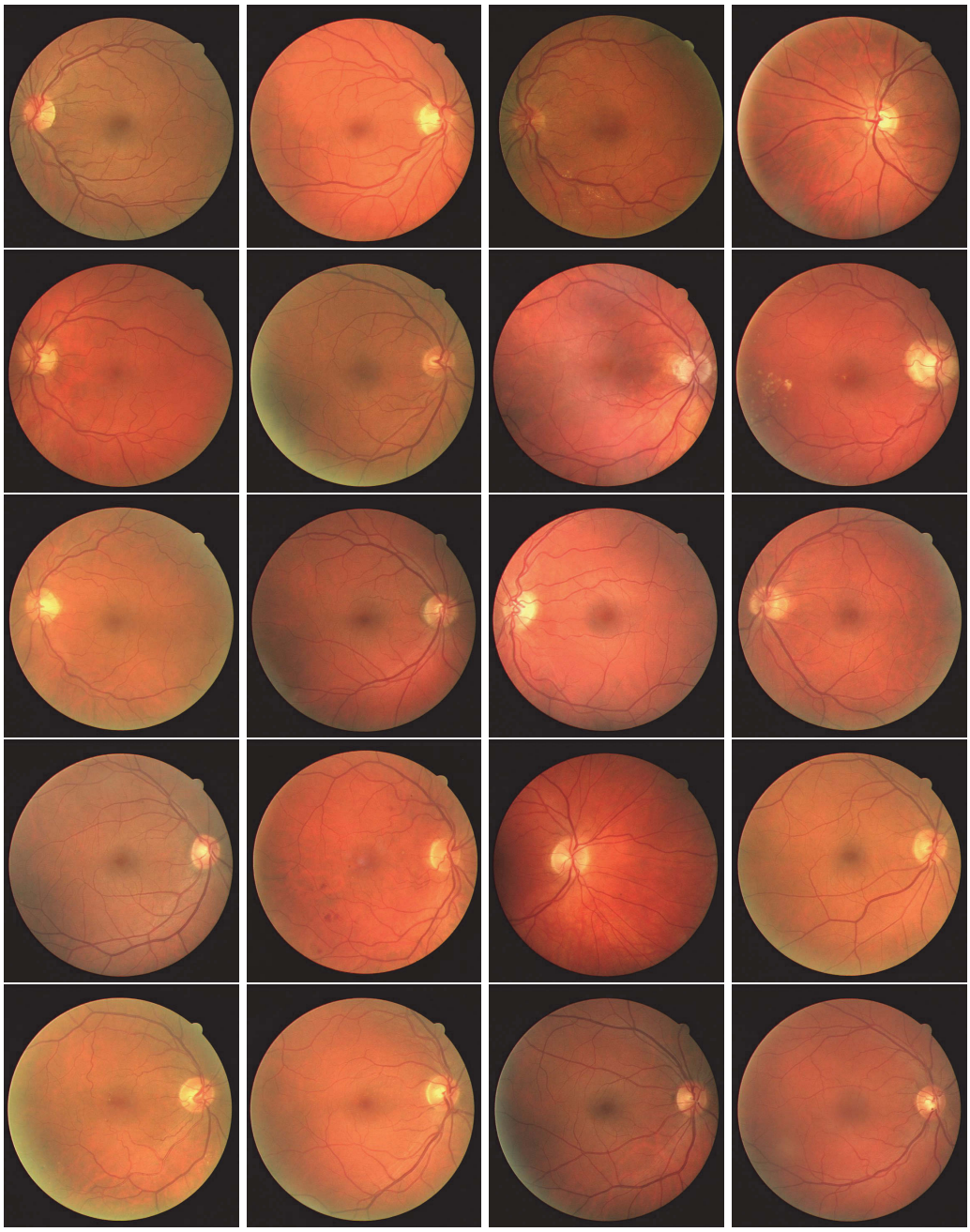


FIGURE 6.16: Fundus Images (1–20) from DRIVE [80] arranged left to right, top to bottom.

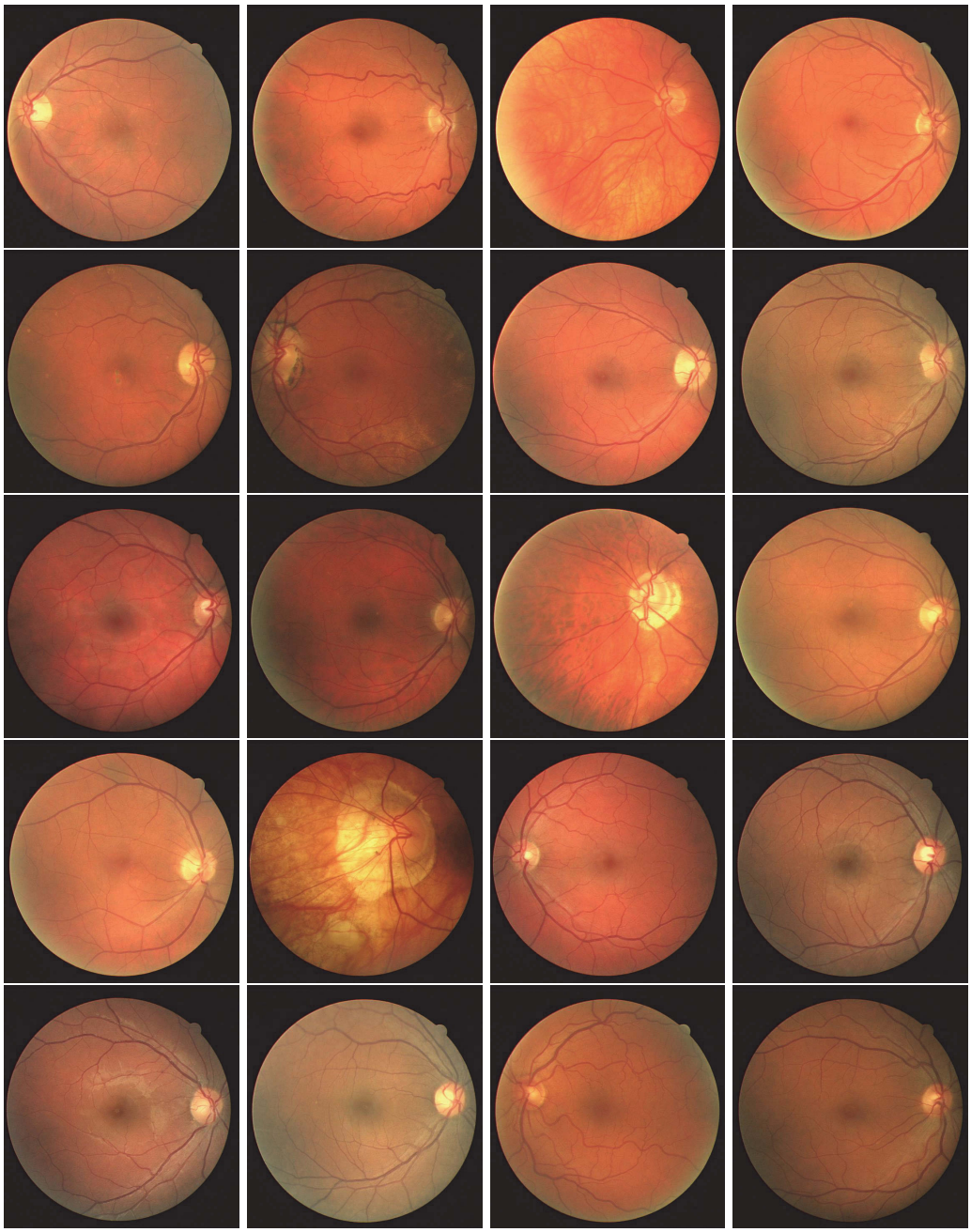


FIGURE 6.17: Fundus Images (21–40) from DRIVE [80] arranged left to right, top to bottom.

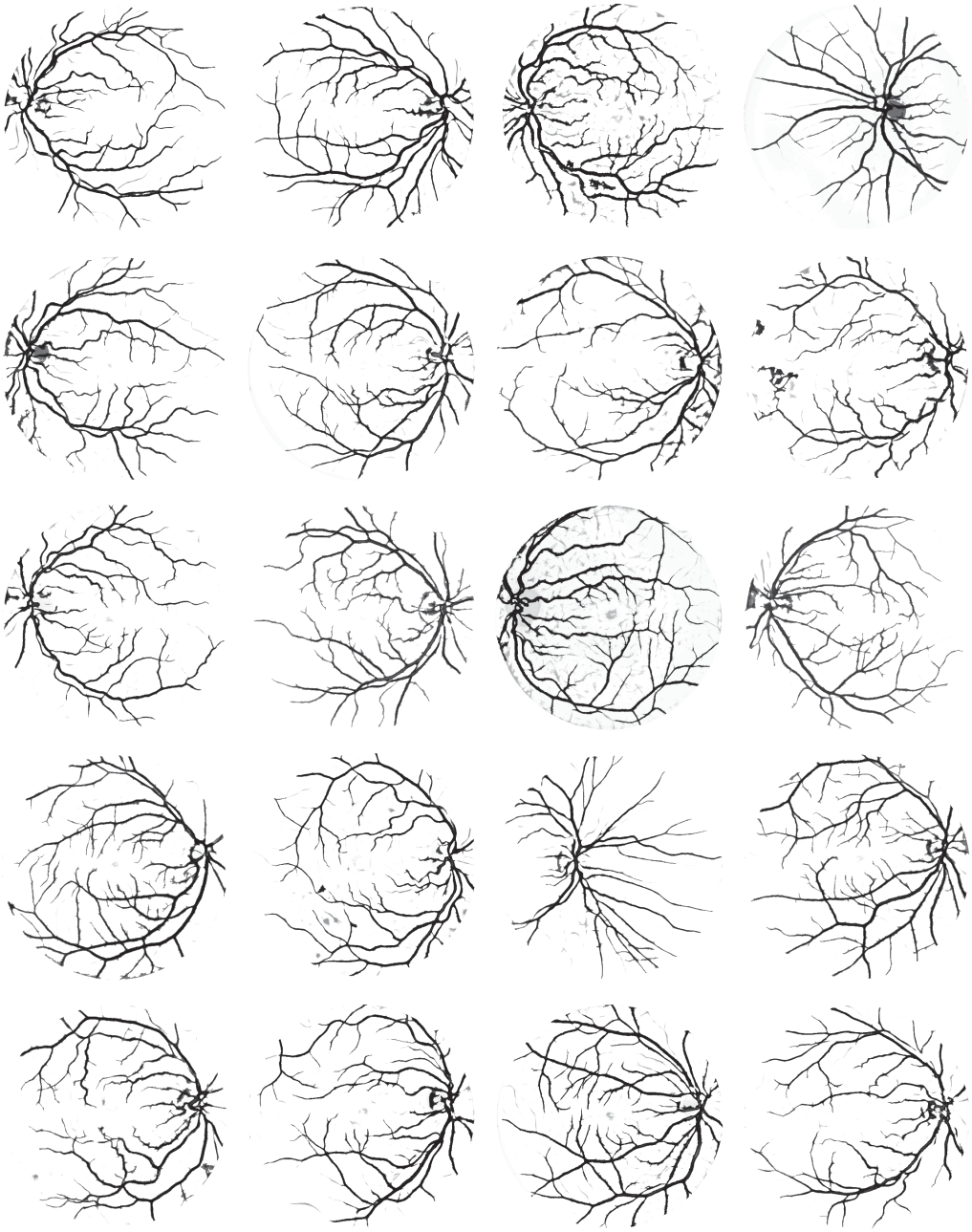


FIGURE 6.18: Fuzzy segmentation results (gray) + detection mask (black) (1–20), arranged left to right, top to bottom.

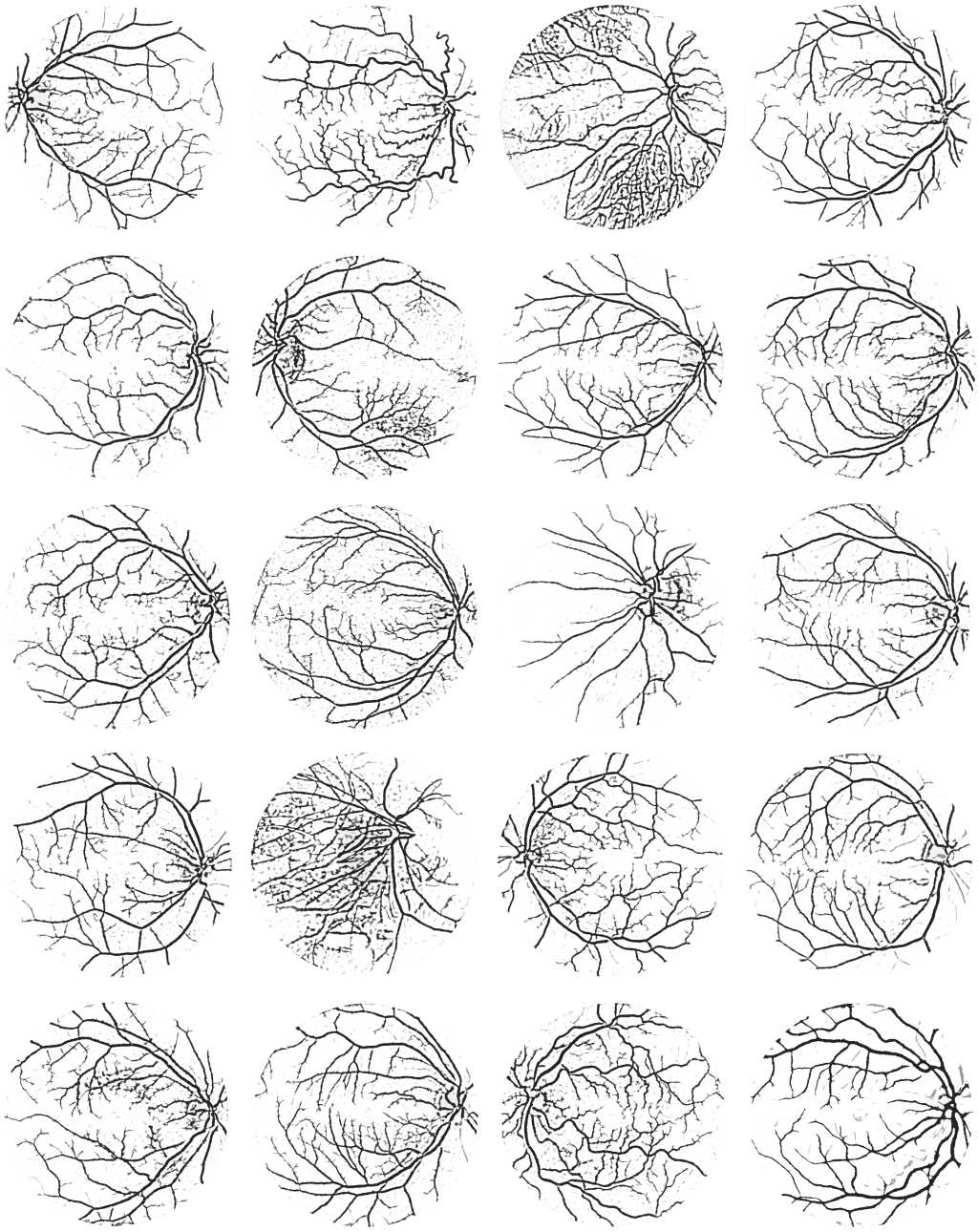


FIGURE 6.19: Fuzzy segmentation results (gray) + detection mask (black) (21–40), arranged left to right, top to bottom.

TABLE 6.3: Fundus Image Clustering Results Summary on the DRIVE database, appended with the results in [52, 57]. Values are expressed in its mean (standard deviation).

Method	Accuracy	L.Acc	Kappa	AUC	Time(s)
Proposed Method (unsupervised)	**0.9558 (0.0051)**	**0.8543**	**0.7736**	**0.9522**	**Total: 30.7 (3.7)**
					Feature Extr.: 17.1 (1.7)
					ERCE+CE: 13.6 (3.3)
Human observer	0.9473 (0.0048)	n/a	0.7589	n/a	7200
Staal (supervised) [80]	0.9442 (0.0065)	0.7749	0.7345	0.9520	900
Niemeijer (supervised) [52]	0.9416 (0.0065)	0.7562	0.7145	0.9294	n/a
Nguyen (unsupervised) [57]	0.9407	0.7883	n/a	n/a	2.5
Zana (unsupervised) [53]	0.9377 (0.0077)	0.7318	0.6971	0.8984	< 180
Al-Diri (supervised) [170]	0.9258 (0.0126)	n/a	0.6716	n/a	660 (180)
Jiang (unsupervised) [171]	0.9212 (0.0076)	0.6915	0.6399	0.9114	n/a
Martínez-Pérez (unsupervised) [172]	0.9316	0.7670	0.6389	n/a	n/a
Chaudhuri (unsupervised) [173]	0.8773 (0.0232)	0.5587	0.3357	0.7878	n/a
All background	0.8727 (0.0123)	n/a	0	n/a	n/a

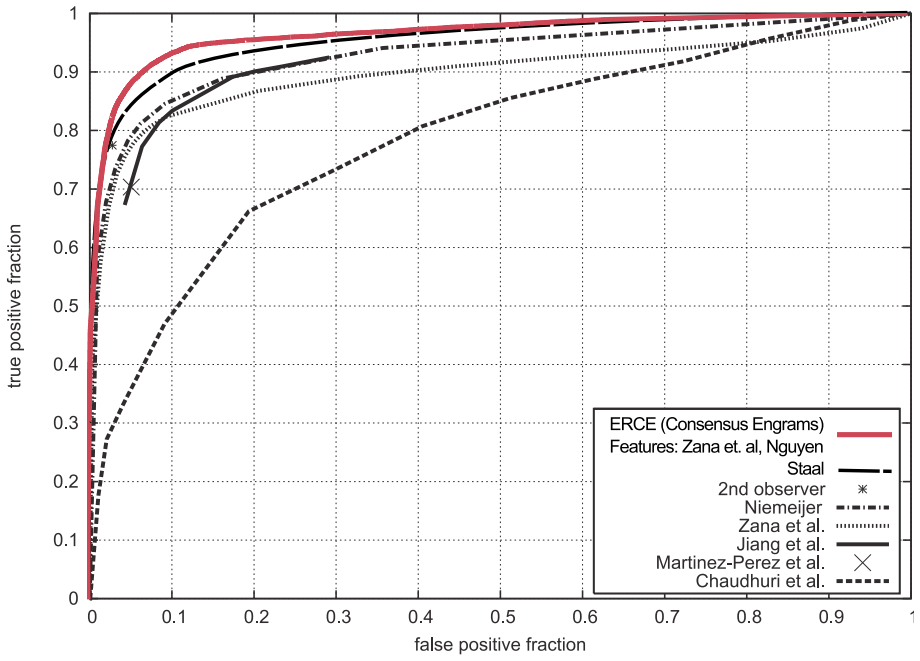


FIGURE 6.20: The ROC curve of the proposed method vs other methods in the literature [80].

Another important characteristic of the algorithm is its relatively lightweight nature. The proposed method requires 30.7 ± 3.7 seconds (17.1 seconds were spent for feature extraction) to extract blood vessels from a 584×565 pixels retinal fundus image, which is significantly faster than Staal's method which takes 15 minutes for each image on average [80].

It should be noted that the performance of the proposed method depends heavily on the feature extraction method. In this implementation, our approach suffers from limitations in Zana's [53] and Nguyen's [57] approaches. Using these features the ERCE+CE still struggles when dealing with noisy images such as image 23 and 34. Future works will include the incorporation of various additional morphological features such as those described by Staal [80].

6.5 Conclusion

Ensemble RCE (ERCE) [55] was proposed in 2014 as an ensemble extension to RCE, improving RCE in terms of memory and computational efficiency, capability to automatically determine the number of clusters, and gracefully handle non-convex datasets in quasilinear complexity. Using the self-evolving cooperative semi-stochastic swarm ERCE constructs the essential building blocks required for large scale consensus clustering using minimal memory and computational resources. Benefiting from the robustness of swarm intelligence, the versatility of voronoi tessellation and the flexibility of graph algorithms, the ERCE is designed to discover natural groupings in both convex and non-convex data. Two strategies are proposed to increase the ERCE swarm diversity including charged particles and self evolution. The complexity of the algorithm has been theoretically and empirically analyzed where we discover that the algorithm has quasilinear complexity in both time and space.

Inspired by Hebb's theories on cell assembly [78], we arrange the particles in the swarm in a fully connected construction called the consensus engrams. This approach substantially increases the scalability of the swarm to handle data whose total size exceeds the physical random access memory (RAM) limit of the machine. The method is specifically designed to extract statistical structure from incomplete data of large volume and noise content. It is insensitive to neither the volume nor the order of data and the engrams only need to "see" just enough representative data to learn the overall statistical structure. The algorithm has been tested on image processing applications with promising results.

On color image segmentation, ERCE with Consensus Engrams (CE) were the leanest in terms of both memory and computational complexity. Both the proposed algorithm and ERCE + CA-tree

+ fWEAC achieved linear complexity in both time and space, with ERCE + CE being significantly leaner.

On the fundus image segmentation application, our proposed unsupervised image segmentation method is able to achieve the highest classification performance compared to the competing methods, which is up to 95.58% global accuracy, 85.43% local accuracy, and 0.7736 kappa score using the fundus image data from the DRIVE database. The method takes about 30 seconds per image which is significantly faster than the method described in [80] (15 minutes).

Chapter 7

Conclusion

This thesis analyzes in great depth, both theoretically and empirically, the development stages, strengths and limitations of our proposed semi-stochastic swarm clustering algorithm, the Rapid Centroid Estimation (RCE) and its variants [4, 55, 71–73]. This thesis establishes the theoretical foundation of the algorithm, much extending those that have been previously proposed in the published manuscripts. The analyses focus mainly on the stability, convergence, computational and memory complexities as well as the overall effectiveness of the algorithm on practical problems.

As a general purpose clustering algorithm, the practical applications of the proposed algorithm are vast. The application scenarios included in this thesis are meant as investigative medium intended for benchmarking the strengths and limitations of the proposed algorithm against other approaches in the literature.

7.1 Primary Findings

1. **Challenges in Particle Swarm Optimization Clustering**

   The research develops from Section 3.3 where we discover the rationale behind the performance degradation of Van Der Merwe - Engelbrecht's Particle Swarm Optimization (PSO) Clustering [83] for problems of higher dimension. The efficiency of PSO clustering degrades exponentially for each dimensionality increment as particle vectors become perpendicular to the maximum likelihood vector. The empirical validation in Table 3.1 provides strong statistical evidence on the inferiority of the algorithm to Stuart Lloyd's k-means in problems over 15 dimensions.

2. **Challenges in the Particle Swarm Clustering Algorithm**

   Cohen and de Castro [75] proposes a new concept of particle-data interaction in their Particle Swarm Clustering (PSC) algorithm. We discover significant issues with their proposition regarding stability, movement redundancies, local convergence, as well as its memory and computational complexity. Theorem 4.1 especially shows how the PSC generalizes to the Expectation Maximization (EM) algorithm — more specifically, k-means — when the stability constraints in Equation 4.37 are satisfied. This theoretical proof completes the empirical observation by Szabo et al. [85], where they stated that *"there were no much gain obtained with the PSC when compared with a standard self-organizing clustering methods"* [85].

   We also discover that all PSC families, including the original Cohen - de Castro's PSC [75], Szabo's Modified memoryless Particle Swarm Clustering (mPSC) [74], and Szabo's Fuzzy Particle Swarm Clustering (FPSC) [70], share the behavior depicted in Figure 4.4 and Figure 4.5. The three algorithms, although proposed under varying names and time frames, are conceptual duplicates of the original PSC.

3. **Rapid Centroid Estimation and its Variants**

   The RCE and its variants in 2012 (RCE 2012) to alleviate general clustering issues pertaining reliability due to initialization and local convergence [4, 71–73]. Using the openly available benchmark datasets from UC Irvine [84] we investigate the performance of the algorithm against its predecessors. The results, presented in Table 5.5, reveals the superior performance the RCE 2012 variants to that of its predecessors. The levels of reliability and repeatability of RCE variants in most cases are substantially greater than those of the other algorithms. Table 5.6 shows that the new algorithms are significantly faster than the predecessors.

4. **Ensemble Rapid Centroid Estimation**

   The discovery of both Equation 4.37 and Lemma 5.2.1 lead to the simplification scheme in the 2014 proposition which further reduce the complexity of RCE and extends its applicability to larger datasets and access to scalable consensus/ensemble approaches which allow it to cope with arbitrarily shaped clusters and estimate the number of clusters in quasilinear time and space complexity. We refer the improved algorithm with a new name called the Ensemble RCE (ERCE). To our knowledge, to date we are the first multi-swarm based ensemble clustering algorithm that has achieved such low complexity in both time and space.

   In this thesis we introduce the self-evolution to the ERCE which is proposed to address the issues in the concept of charged particles [55]. Self evolution allows the RCE to have unlimited number of particles in order to satisfy the cluster entropy requirement. The approach further reduces the memory requirement of ERCE.

5. **Consensus Engrams**

   The consensus engrams further simplifies the ERCE to deal with partial data. Instead of dealing with the complete consensus matrix, the consensus engrams encodes the feature statistical properties in the form of relationship between subswarms and particles. The algorithm was applied on image clustering and retinal fundus images with satisfactory result. Based on the empirical experiments, the proposed approach provides significant improvement to ERCE in term of computational and memory efficiency as shown in Figure 6.12. In retinal fundus vessel extraction task, Table 6.3 shows that the algorithm outperforms even Staal's supervised approach [80] with much lower computational complexity.

7.2 Limitations of the research

1. **Sensitivity to feature quality**

   RCE variants are limited on performing unsupervised clustering on problems which classes are separable based on the statistical landscape of the feature space. Should the majority of the features contain redundant and irrelevant information, the algorithm would experience a significant performance degradation.

2. **Sensitivity to distance quantifier selection**

   The performance of RCE variants are dependent on the selection of quantifier which defines the distance between two feature vectors. The definition of "distance" will need to be pre-specified by the user.

3. **Final result is nondeterministic**

   Like other clustering algorithms, the results of RCE varies on each repetitions. The proposed strategies (particle reset, substitution, swarm) decrease the likelihood of the particles getting trapped in a local minimum, hence increasing the likelihood of getting an optimal partition. The chance of returning suboptimal partitions, although minimized, still exist.

4. **Sensitivity to the consensus/ensemble aggregation or graph partitioning method**

   The ERCE requires the user to manually specify the graph partitioning algorithm. The user needs to choose between clustering based on close connectivity (e.g. normalized cuts) or arbitrary interconnected shapes (e.g. single linkage), but not both.

5. **Sensitivity to the choice of parameters**

The choice of parameters, especially the width of fuzzifier, number of particles, swarm size, substitution probability, and stagnation threshold (particle reset), are significant parameters that determine the performance of the algorithm. The penalty factor for cross-cluster closeness (consensus engrams) and the variations of the data fed to the engrams would determine whether the synaptic bonds learnt by the engrams can properly represent the statistical structure of the feature space.

In a highly non-convex problems, there need to be enough particle in order to ensure proper extraction of the statistical structure of the data. However, the only parameter that RCE uses to control the growth of the particle is the cluster entropy, which depends on the user-defined width of the fuzzifier.

6. **RCE does not perform supervised learning**

   The RCE belongs to the class of unsupervised clustering algorithm. It is designed to minimize the likelihood function given the data and the assumption of the model. Supervised learning such as Neural Networks and Support Vector Machines belong to the different class of machine learning which are specifically intended for function approximation. For such task, the RCE may not be the right tool.

7.3 Recommendations for Future Work

The RCE algorithm has been under continuous improvement since its proposal in 2012 [71, 73]. Based on the aforementioned limitations, the algorithm is still far from finishing. One of the major goals includes making the RCE parameter free. As the algorithm is designed for parallel processing, a possible future work is the parallelization of RCE for mining very large data. Finally further practical implementations and empirical tests on even larger and more complex data needs to be done.

Bibliography

[1] P. Dayan, M. Sahani, and G. Deback. Unsupervised learning. In *In The MIT Encyclopedia of the Cognitive Sciences*. The MIT Press, 1999.

[2] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letter*, 31(8): 651–666, June 2010. ISSN 0167-8655. doi: 10.1016/j.patrec.2009.09.011.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999. ISSN 0360-0300. doi: 10.1145/331499.331504.

[4] M. Yuwono, S. Su, B. Moulton, and H. Nguyen. Data clustering using variants of rapid centroid estimation. *IEEE Transactions on Evolutionary Computation*, 18(3):366–377, June 2014. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281545.

[5] W. Li, L. Jaroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, Mar 2001.

[6] W. Li, L. Fu, B. Niu, S. Wu, and J. Wooley. Ultrafast clustering algorithms for metagenomic sequence analysis. *Briefings in Bioinformatics*, 13(6):656–668, Nov 2012.

[7] C. P. Cheng, Y. C. Liu, Y. L. Tsai, and V. S. Tseng. An efficient method for mining cross-timepoint gene regulation sequential patterns from time course gene expression datasets. *BMC Bioinformatics*, 14 Suppl 12:S3, 2013.

[8] Y. Iwasaki, K. Wada, Y. Wada, T. Abe, and T. Ikemura. Notable clustering of transcription-factor-binding motifs in human pericentric regions and its biological significance. *Chromosome Res.*, 21(5):461–474, Aug 2013.

[9] W. Chen, Y. Cheng, C. Zhang, S. Zhang, and H. Zhao. MSClust: A Multi-Seeds based Clustering algorithm for microbiome profiling using 16S rRNA sequence. *J. Microbiol. Methods*, 94(3):347–355, Sep 2013.

[10] P. Kolekar, M. Kale, and U. Kulkarni-Kale. Alignment-free distance measure based on return time distribution for sequence analysis: applications to clustering, molecular phylogeny and subtyping. *Mol. Phylogenet. Evol.*, 65(2):510–522, Nov 2012.

[11] C. K. Kontos and A. Scorilas. Molecular cloning of novel alternatively spliced variants of BCL2L12, a new member of the BCL2 gene family, and their expression analysis in cancer cells. *Gene*, 505(1):153–166, Aug 2012.

[12] M. M. Rashid, M. R. Karim, B. S. Jeong, and H. J. Choi. Efficient mining of interesting patterns in large biological sequences. *Genomics Inform*, 10(1):44–50, Mar 2012.

[13] A. Balasubramanian, B. Prabhakaran, and A. Sawant. Mining pattern sequences in respiratory tumor motion data. *Conf Proc IEEE Eng Med Biol Soc*, 2012:5262–5265, 2012.

[14] M. Yuwono, B. D. Moulton, S. W. Su, B. G. Celler, and H. T. Nguyen. Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *Biomedical Engineering Online*, 11:9, 2012. [PubMed Central:PMC3395835] [DOI:10.1186/1475-925X-11-9] [PubMed:22336100].

[15] M. Yuwono, A. M. Handojoseno, and H. T. Nguyen. Optimization of head movement recognition using Augmented Radial Basis Function Neural Network. *Conf Proc IEEE Eng Med Biol Soc*, 2011:2776–2779, 2011. [DOI:10.1109/IEMBS.2011.6090760] [PubMed:22254917].

[16] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Gait episode identification based on wavelet feature clustering of spectrogram images. *Conf Proc IEEE Eng Med Biol Soc*, 2012:2949–2952, 2012. [DOI:10.1109/EMBC.2012.6346582] [PubMed:23366543].

[17] M. Yuwono, S. W. Su, Y. Guo, B. D. Moulton, and H. T. Nguyen. Unsupervised nonparametric method for gait analysis using a waist-worn inertial sensor. *Applied Soft Computing*, 14, Part A(0):72 – 80, 2014. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2013.07.027. URL `http://www.sciencedirect.com/science/article/pii/S1568494613002822`. Special issue on hybrid intelligent methods for health technologies.

[18] U. Abdulla, K. Taylor, M. Barlow, and K. Naqshbandi. Measuring walking and running cadence using magnetometers. In *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2013*, pages 1458–1462, July 2013. doi: 10.1109/TrustCom.2013.176.

[19] A. Oliynyk, C. Bonifazzi, F. Montani, and L. Fadiga. Automatic online spike sorting with singular value decomposition and fuzzy C-mean clustering. *BMC Neurosci*, 13:96, 2012. [PubMed Central:PMC3473300] [DOI:10.1186/1471-2202-13-96] [PubMed:22871125].

[20] C. C. Lin, C. H. Lee, C. S. Fuh, H. F. Juan, and H. C. Huang. Link clustering reveals structural characteristics and biological contexts in signed molecular networks. *PLoS ONE*, 8(6):e67089, 2013. [PubMed Central:PMC3691148] [DOI:10.1371/journal.pone.0067089] [PubMed:23826198].

[21] J. Wang, P. Liu, M. F H She, S. Nahavandi, and A. Kouzani. Biomedical time series clustering based on non-negative sparse coding and probabilistic topic model. *Comput Methods Programs Biomed*, 111(3):629–641, Sep 2013. [DOI:10.1016/j.cmpb.2013.05.022] [PubMed:23846155].

[22] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Gait cycle spectrogram analysis using a torso-attached inertial sensor. *Conf Proc IEEE Eng Med Biol Soc*, 2012:6539–6542, 2012. [DOI:10.1109/EMBC.2012.6347492] [PubMed:23367427].

[23] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Unsupervised segmentation of heel-strike IMU data using rapid cluster estimation of wavelet features. *Conf Proc IEEE Eng Med Biol Soc*, 2013:953–956, 2013. [DOI:10.1109/EMBC.2013.6609660] [PubMed:24109847].

[24] M. Yuwono, J. Sepulveda, and A. M. Ardi Handojoseno. Centroid extraction from Hartmann-Shack images using swarm clustering approach. *Conf Proc IEEE Eng Med Biol Soc*, 2012: 1446–1449, 2012. [DOI:10.1109/EMBC.2012.6346212] [PubMed:23366173].

[25] M. Yuwono. Unwrapping hartmann-shack images of off-axis aberration using artificial centroid injection method. In *IEEE Conference in Biomedical Engineering and Informatics (BMEI)*, pages 560–564, 2011.

[26] W. Cui, Y. Wang, Y. Fan, Y. Feng, and T. Lei. Localized FCM Clustering with Spatial Information for Medical Image Segmentation and Bias Field Estimation. *Int J Biomed Imaging*, 2013:930301, 2013. [PubMed Central:PMC3749607] [DOI:10.1155/2013/930301] [PubMed:23997761].

[27] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2D and 3D images. *PLoS ONE*, 8(8):e71715, 2013. [PubMed Central:PMC3748125] [DOI:10.1371/journal.pone.0071715] [PubMed:23977123].

[28] F. Mualla, S. Scholl, B. Sommerfeldt, A. Maier, and J. Hornegger. Automatic Cell Detection in Bright-field Microscope Images Using SIFT, Random Forests, and Hierarchical Clustering. *IEEE Trans Med Imaging*, Aug 2013. [DOI:10.1109/TMI.2013.2280380] [PubMed:24001988].

[29] A. Abdullah, A. Hirayama, S. Yatsushiro, M. Matsumae, and K. Kuroda. Cerebrospinal fluid image segmentation using spatial fuzzy clustering method with improved evolutionary Expectation Maximization. *Conf Proc IEEE Eng Med Biol Soc*, 2013:3359–3362, 2013. [DOI:10.1109/EMBC.2013.6610261] [PubMed:24110448].

[30] T. Alexandrov, I. Chernyavsky, M. Becker, F. von Eggeling, and S. Nikolenko. Analysis and interpretation of imaging mass spectrometry data by clustering mass-to-charge images according to their spatial similarity. *Anal. Chem.*, 85(23):11189–11195, Dec 2013. [DOI:10.1021/ac401420z] [PubMed:24180335].

[31] J. Dietlmeier, O. Ghita, H. Duessmann, J. H. Prehn, and P. F. Whelan. Unsupervised mitochondria segmentation using recursive spectral clustering and adaptive similarity models. *J. Struct. Biol.*, 184(3):401–408, Dec 2013. [DOI:10.1016/j.jsb.2013.10.013] [PubMed:24184470].

[32] M. Hassan, A. Chaudhry, A. Khan, and M. A. Iftikhar. Robust information gain based fuzzy c-means clustering and classification of carotid artery ultrasound images. *Comput Methods Programs Biomed*, 113(2):593–609, Feb 2014. [DOI:10.1016/j.cmpb.2013.10.012] [PubMed:24239296].

[33] A. Faktor and M. Irani. "Clustering by Composition" - Unsupervised Discovery of Image Categories. *IEEE Trans Pattern Anal Mach Intell*, Dec 2013. [DOI:5F856FDE-10D1-46A6-98B9-0A0053CD27C1] [PubMed:24344078].

[34] N. E. El Harchaoui, M. Ait Kerroum, A. Hammouch, M. Ouadou, and D. Aboutajdine. Unsupervised approach data analysis based on fuzzy possibilistic clustering: application to medical image MRI. *Comput Intell Neurosci*, 2013:435497, 2013. [PubMed Central:PMC3891616] [DOI:10.1155/2013/435497] [PubMed:24489535].

[35] C. H. Lyoo, P. Zanotti-Fregonara, S. S. Zoghbi, J. S. Liow, R. Xu, V. W. Pike, C. A. Zarate, M. Fujita, and R. B. Innis. Image-derived input function derived from a supervised clustering algorithm: methodology and validation in a clinical protocol using [11C](R)-rolipram. *PLoS ONE*, 9(2):e89101, 2014. [PubMed Central:PMC3930688] [DOI:10.1371/journal.pone.0089101] [PubMed:24586526].

[36] B. C. Ko and J. Y. Nam. Object-of-interest image segmentation based on human attention and semantic region clustering. *J Opt Soc Am A Opt Image Sci Vis*, 23(10):2462–2470, Oct 2006. [PubMed:16985531].

[37] T. H. Tsai, W. H. Cheng, C. W. You, M. C. Hu, A. W. Tsui, and H. Y. Chi. Learning and recognition of on-premise signs from weakly labeled street view images. *IEEE Trans Image Process*, 23(3):1047–1059, Mar 2014. [DOI:10.1109/TIP.2014.2298982] [PubMed:24474374].

[38] B. Jun, I. Choi, and D. Kim. Local transform features and hybridization for accurate face and human detection. *IEEE Trans Pattern Anal Mach Intell*, 35(6):1423–1436, Jun 2013. [DOI:10.1109/TPAMI.2012.219] [PubMed:23599056].

[39] B. Yao, Z. Liu, X. Nie, and S. C. Zhu. Animated Pose Templates for Modelling and Detecting Human Actions. *IEEE Trans Pattern Anal Mach Intell*, Aug 2013. [DOI:5DD67F8B-FF2B-4ED7-8C16-957691F64B8E] [PubMed:23917419].

[40] W. Deng, J. Hu, J. Lu, and J. Guo. Transform-Invariant PCA: A Unified Approach to Fully Automatic Face Alignment, Representation, and Recognition. *IEEE Trans Pattern Anal Mach Intell*, Oct 2013. [DOI:AC37C0FF-FF8D-456D-9D9E-36CF42BFEB14] [PubMed:24101334].

[41] B. Z. Yao, B. X. Nie, Z. Liu, and S. C. Zhu. Animated pose templates for modeling and detecting human actions. *IEEE Trans Pattern Anal Mach Intell*, 36(3):436–452, Mar 2014. [DOI:10.1109/TPAMI.2013.144] [PubMed:24457502].

[42] A. B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Trans Pattern Anal Mach Intell*, 30(5):909–926, May 2008. [DOI:10.1109/TPAMI.2007.70738] [PubMed:18369258].

[43] Merriam-Webster Online Dictionary. Online, 2014. URL `http://www.merriam-webster.com`.

[44] T. G. Dietterich. Ensemble methods in machine learning. In I. J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, New York, 2000.

[45] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver, editors, *ICML Unsupervised and Transfer Learning*, volume 27 of *JMLR Proceedings*, pages 37–50. JMLR.org, 2012. URL `http://dblp.uni-trier.de/db/journals/jmlr/jmlrp27.html#Baldi12`.

[46] S. Dasgupta. Algorithms for minimally supervised learning. Online, November 9 2013. URL `http://cseweb.ucsd.edu/~dasgupta/papers/paris.pdf`.

[47] F. Berman, G. Fox, and T. Hey. *The Data Deluge: An e-Science Perspective*, chapter 36, pages 809–824. John Wiley & Sons, Ltd, 2003. ISBN 9780470867167. doi: 10.1002/0470867167.ch36. URL `http://dx.doi.org/10.1002/0470867167.ch36`.

[48] R. Bellazzi, M. Diomidous, I. N. Sarkar, K. Takabayashi, A. Ziegler, and A. T. McCray. Data analysis and data mining: current issues in biomedical informatics. *Methods Inf Med*, 50(6): 536–544, 2011.

[49] A. D. Corlan. Medline trend: automated yearly statistics of pubmed results for any query, 2004. Online, 2014. URL `http://dan.corlan.net/medline-trend.html`.

[50] M. Iacomi, D. Cascio, F. Fauci, and G. Raso. Mammographic images segmentation based on chaotic map clustering algorithm. *BMC Med Imaging*, 14:12, 2014. [PubMed Central:PMC3987162] [DOI:10.1186/1471-2342-14-12] [PubMed:24666766].

[51] Y. Zhu, F. Li, T. J. Vadakkan, M. Zhang, J. Landua, W. Wei, J. Ma, M. E. Dickinson, J. M. Rosen, M. T. Lewis, M. Zhan, and S. T. Wong. Three-dimensional vasculature reconstruction of tumour microenvironment via local clustering and classification. *Interface Focus*, 3(4):20130015, Aug 2013. [PubMed Central:PMC3915834] [DOI:10.1098/rsfs.2013.0015] [PubMed:24511379].

[52] M. Niemeijer, J. Staal, B. van Ginneken, M. Loog, and M. D. Abramoff. Comparative study of retinal vessel segmentation methods on a new publicly available database. In *SPIE Medical Imaging*, volume 5370, pages 648–656, 2004. doi: 10.1117/12.535349. URL `http://dx.doi.org/10.1117/12.535349`.

[53] F. Zana and J.-C. Klein. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Transactions on Image Processing*, 10(7):1010–1019, Jul 2001. ISSN 1057-7149. doi: 10.1109/83.931095.

[54] D. Liang, Z. YongPing, and Z. Xueying. An approach to retinal image segmentations using fuzzy clustering in combination with morphological filters. In *30th Chinese Control Conference (CCC) 2011*, pages 3062–3065, July 2011.

[55] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. An algorithm for scalable clustering: Ensemble rapid centroid estimation. In *Proc of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2014.

[56] professional vision care. Eye health. Online, 2014. URL `http://www.professionalvisioncareinc.com/vision-care-services/eye-health-for-Westerville-Johnstown-OH-New-Albany-Gahanna-Worthington-&-Sunbury`.

[57] U. T. Nguyen, A. Bhuiyan, L. A. Park, and K. Ramamohanarao. An effective retinal blood vessel segmentation method using multi-scale line detection. *Pattern Recognition*, 46(3):703

– 715, 2013. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2012.08.009. URL http://www.sciencedirect.com/science/article/pii/S003132031200355X.

[58] G. Wewerka and B. Iglseder. [Measuring gait velocity in the elderly with a gait analysis system and a 10-meter walk test : A comparison.]. *Zeitschrift fur Gerontologie und Geriatrie*, Dec 2013.

[59] M. Yuwono, S. W. Su, and B. D. Moulton. Fall detection using a gaussian distribution of clustered knowledge, augmented radial basis neural-network, and multilayer perceptron. In *6th International Conference on Broadband and Biomedical Communications (IB2Com), 2011*, pages 145–150, Nov 2011. doi: 10.1109/IB2Com.2011.6217909.

[60] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The Computer Journal*, 9(4):373–380, 1967. doi: 10.1093/comjnl/9.4.373. URL http://comjnl.oxfordjournals.org/content/9/4/373.abstract.

[61] F. R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, Americal Mathematical Society*, 1997.

[62] J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76, 2007. ISSN 1089-778X. doi: 10.1109/TEVC.2006.877146.

[63] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[64] A. L. Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):835–850, Jun 2005.

[65] F. Duarte, A. L. N. Fred, A. Lourenco, and M. Rodrigues. Weighting cluster ensembles in evidence accumulation clustering. In *Portuguese conference on Artificial intelligence, 2005. epia 2005.*, pages 159–167, 2005. doi: 10.1109/EPIA.2005.341287.

[66] T. Wang. Ca-tree: A hierarchical structure for efficient and scalable coassociation-based cluster ensembles. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(3):686–698, 2011. ISSN 1083-4419. doi: 10.1109/TSMCB.2010.2086059.

[67] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka. On the comparison of relative clustering validity criteria. In *SIAM International Conference on Data Mining*, pages 733–744, 2009.

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

[69] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, December 2010.

[70] A. Szabo, L. de Castro, and M. Delgado. The proposal of a fuzzy clustering algorithm based on particle swarm. In *Proc. of the Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 459–465, oct. 2011.

[71] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Fast unsupervised learning method for rapid estimation of cluster centroids. In *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, pages 889–896, June 10–15 2012.

[72] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Optimization strategies for rapid centroid estimation. In *Proc. of the 34rd Annual International Conference of the IEEE EMBS*, pages 6212–6215, San Diego, Aug. 28–sept. 1 2012.

[73] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen. Method for increasing the computation speed of an unsupervised learning approach for data clustering. In *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, pages 2957–2964, June 10–15 2012.

[74] A. Szabo, A. K. F. Prior, and L. N. de Castro. The proposal of a velocity memoryless clustering swarm. In *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, pages 1–5, Barcelona, July 18–23 2010.

[75] S. C. M. Cohen and L. N. de Castro. Data clustering with particle swarms. In *Proc. of the 2006 IEEE Congress on Evolutionary Computation*, pages 1792–1798, Vancouver, 2006.

[76] T. Wang. Comparing hard and fuzzy c-means for evidence-accumulation clustering. In *Proceedings of the 18th International Conference on Fuzzy Systems*, FUZZ-IEEE'09, pages 468–473, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3596-8.

[77] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. of the 6th International Symposium on Micro Machine and Human Science*, pages 39–43, Oct. 4–6 1995.

[78] D. O. Hebb. *Organization of Behavior*. Wiley, 1949.

[79] Lotus Hill Reseach Institute for Computer Vision and Information Science. Image parsing. Online, 2007. URL `http://www.imageparsing.com/`.

[80] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4): 501–509, April 2004. ISSN 0278-0062. doi: 10.1109/TMI.2004.825627.

[81] M. Jiang, Y. Luo, and S. Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102 (1):8 – 16, 2007. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/j.ipl.2006.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0020019006003103`.

[82] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1): 58–73, Feb 2002. ISSN 1089-778X. doi: 10.1109/4235.985692.

[83] D. W. Van Der Merwe and A. Engelbrecht. Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 1, pages 215–220 Vol.1, Dec 2003. doi: 10.1109/CEC.2003.1299577.

[84] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[85] A. Szabo, A. Prior, and L. de Castro. The behavior of particles in the particle swarm clustering algorithm. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–7, July 2010. doi: 10.1109/FUZZY.2010.5584118.

[86] A. Lourenço, S. Rota Bulò, A. Fred, and M. Pelillo. Consensus clustering with robust evidence accumulation. In A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 8081 of *Lecture Notes in Computer Science*, pages 307–320. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40394-1. doi: 10.1007/978-3-642-40395-8_23.

[87] J. C. Bezdek. Mathematical models for systematic and taxonomy. In *Estabrook, G. (Ed.), Proc. 8th International Conference on Numerical Taxonomy*, pages 143–166, Freeman, San Fransisco, CA, 1975.

[88] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 03 1951. doi: 10.1214/aoms/1177729694. URL `http://dx.doi.org/10.1214/aoms/1177729694`.

[89] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001. ISSN 1559-1662. doi: 10.1145/584091.584093. URL `http://doi.acm.org/10.1145/584091.584093`.

[90] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000. ISSN 0162-8828. doi: 10.1109/34.868688.

[91] A. L. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Proc. of the International Conference on Pattern Recognition*, pages 276–280, 2002.

[92] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1-2):91–118, 2003. ISSN 0885-6125. doi: 10.1023/A:1023949509487. URL `http://dx.doi.org/10.1023/A%3A1023949509487`.

[93] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23 – 63, 1987. ISSN 0364-0213. doi: http://dx.doi.org/10.1016/S0364-0213(87)80025-3. URL `http://www.sciencedirect.com/science/article/pii/S0364021387800253`.

[94] G. Carpenter and S. Grossberg. Adaptive resonance theory. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks, Second Edition*, pages 87–90. MIT Press, 2003.

[95] T. Kohonen. Neurocomputing: Foundations of research. chapter Self-organized Formation of Topologically Correct Feature Maps, pages 509–521. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL `http://dl.acm.org/citation.cfm?id=65669.104428`.

[96] E. F. Krause. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Courier Dover Publications, 1986.

[97] MathWorks. kmeans. Online, 2014. URL `http://www.mathworks.com.au/help/stats/kmeans.html`.

[98] R Documentation. hclust {stats}. Online, 2014. URL `http://stat.ethz.ch/R-manual/R-patched/library/stats/html/hclust.html`.

[99] J. Ducci. Derivations for linear algebra and optimization. Online. http://www.cs.berkeley.edu/ jduchi/projects/general_notes.pdf.

[100] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-13360-1.

[101] I. Dagan, L. Lee, and F. Pereira. Similarity-based methods for word sense disambiguation. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 56–63, Stroudsburg, PA, USA, 1997. Association

for Computational Linguistics. doi: 10.3115/979617.979625. URL http://dx.doi.org/10.3115/979617.979625.

[102] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *Proceedings. International Symposium on Information Theory (ISIT), 2004.*, pages 31–, June 2004. doi: 10.1109/ISIT.2004.1365067.

[103] R. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal,*, 29(2):147–160, April 1950. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1950.tb00463.x.

[104] T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biol. Skr.*, 5:1–34, 1948.

[105] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, July 1945. ISSN 0012-9658. doi: 10.2307/1932409. URL http://dx.doi.org/10.2307/1932409.

[106] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

[107] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, March 2003. ISSN 1532-4435. doi: 10.1162/153244303321897735. URL http://dx.doi.org/10.1162/153244303321897735.

[108] M. Meila. Comparing clusterings: an axiomatic view. In L. D. Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML-05)*, pages 577–584, 2005. URL http://www.machinelearning.org/proceedings/icml2005/papers/073_ComparingClustering_Meila.pdf.

[109] Y. Yao. Information-theoretic measures for knowledge discovery and data mining. In Karmeshu, editor, *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, volume 119 of *Studies in Fuzziness and Soft Computing*, pages 115–136. Springer Berlin Heidelberg, 2003. ISBN 978-3-642-05531-7. doi: 10.1007/978-3-540-36212-8_6. URL http://dx.doi.org/10.1007/978-3-540-36212-8_6.

[110] T. O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man and Cybernetics,*, 17(3):517–519, May 1987. ISSN 0018-9472. doi: 10.1109/TSMC.1987.4309069.

[111] S. Dasgupta. *CSE 291: Topics in unsupervised learning — Lecture 2: The k-means problem*. University of California San Diego, Jacobs School of Engineering, Spring 2008.

[112] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[113] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. http://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html.

[114] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. URL `http://dl.acm.org/citation.cfm?id=1283383.1283494`.

[115] 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. doi: 10.1080/01969727308546046. URL `http://dx.doi.org/10.1080/01969727308546046`.

[116] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[117] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning 2^{nd} Edition*. Springer-Verlag, 2009.

[118] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39 (1):1–38, 1977. ISSN 00359246. doi: 10.2307/2984875. URL `http://web.mit.edu/6.435/www/Dempster77.pdf`.

[119] M. A. T. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 24(3):381–396, March 2002. ISSN 0162-8828. doi: 10.1109/34.990138.

[120] L. Xu and M. I. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Comput.*, 8(1):129–151, January 1996. ISSN 0899-7667. doi: 10.1162/neco.1996.8.1.129. URL `http://dx.doi.org/10.1162/neco.1996.8.1.129`.

[121] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook Version: November 15, 2012*. University of Waterloo, Toronto, November 2012. URL `http://orion.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`.

[122] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X.

[123] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

[124] J. C. Bezdek. Cluster validity with fuzzy sets. *Journal of Cybernetics*, 3(3):58–73, 1973. doi: 10.1080/01969727308546047. URL `http://dx.doi.org/10.1080/01969727308546047`.

[125] B. Balasko, J. Abonyi, and B. Feil. *Fuzzy Clustering and Data Analysis Toolbox*. Department of Process Engineering, University of Veszprem, Veszprem, Hungary. URL `http://www.fmt.vein.hu/softcomp/fclusttoolbox/`.

[126] M. R. Rezaee, B. Lelieveldt, and J. Reiber. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3–4):237–246, 1998. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/S0167-8655(97)00168-2. URL `http://www.sciencedirect.com/science/article/pii/S0167865597001682`.

[127] L. M. Silva, C. S. Felgueiras, L. A. Alexandre, and J. Marques de Sá. Error entropy in classification problems: A univariate data analysis. *Neural Comput.*, 18(9):2036–2061, September 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.9.2036. URL `http://dx.doi.org/10.1162/neco.2006.18.9.2036`.

[128] A. Bensaid, L. Hall, J. Bezdek, L. P. Clarke, M. Silbiger, J. Arrington, and R. Murtagh. Validity-guided (re)clustering with applications to image segmentation. *IEEE Transactions on Fuzzy Systems*, 4(2):112–123, May 1996. ISSN 1063-6706. doi: 10.1109/91.493905.

[129] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(0):53 – 65, 1987. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/0377-0427(87)90125-7. URL `http://www.sciencedirect.com/science/article/pii/0377042787901257`.

[130] J. Bezdek and N. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):301–315, 1998. ISSN 1083-4419. doi: 10.1109/3477.678624.

[131] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. ISSN 0176-4268. doi: 10.1007/BF01908075.

[132] R. Xu, J. Xu, and D. Wunsch. A comparison study of validity indices on swarm-intelligence-based clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(4):1243–1256, Aug. 2012. ISSN 1083-4419. doi: 10.1109/TSMCB.2012.2188509.

[133] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974. doi: 10.1080/03610927408827101.

[134] S. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. *Extraction and Analysis of Facebook Friendship Relations*. 2012. URL `http://www.emilio.ferrara.name/wp-content/uploads/2011/06/SN-76.pdf`.

[135] M. Thelwall. Social networks, gender, and friending: An analysis of myspace member profiles. *J. Am. Soc. Inf. Sci. Technol.*, 59(8):1321–1330, June 2008. ISSN 1532-2882. doi: 10.1002/asi.v59:8. URL `http://dx.doi.org/10.1002/asi.v59:8`.

[136] U. Gargi, W. Lu, V. Mirrokni, and S. Yoon. Large-scale community detection on youtube for topic discovery and exploration. In *in Proc. of the Fifth international AAAI Conference on Weblogs and Social Media*, pages 486–489, 2011.

[137] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 407–416, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347176. URL `http://doi.acm.org/10.1145/347090.347176`.

[138] L. Page. Method for node ranking in a linked database, 2001.

[139] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1:385–408, 2004.

[140] R. Morales, T. Di Matteo, and T. Aste. Dependency structure and scaling properties of financial time series are related. *Scientific Report*, 4, 2014. doi: 10.1038/srep04589.

[141] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl 1):S145–S154, 2002. doi: 10.1093/bioinformatics/18.suppl_1.S145. URL `http://bioinformatics.oxfordjournals.org/content/18/suppl_1/S145.abstract`.

[142] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 11(9):1074–1085, November 2006. ISSN 0278-0070. doi: 10.1109/43.159993. URL `http://dx.doi.org/10.1109/43.159993`.

[143] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z. URL `http://dx.doi.org/10.1007/s11222-007-9033-z`.

[144] J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society, Series C (Applied Statistics).*, 18(1):54–64, 1969. URL `http://www.jstor.org/stable/2346439`.

[145] E. A. Dinic. Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation. *Soviet Math Doklady*, 11:1277–1280, 1970. URL `http://www.cs.bgu.ac.il/~{}dinitz/D70.pdf`.

[146] Y. Dinitz. Dinitz' Algorithm: The Original Version and Even's Version. In O. Goldreich, A. Rosenberg, and A. Selman, editors, *Theoretical Computer Science*, volume 3895 of *Lecture Notes in Computer Science*, chapter 10, pages 218–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32880-3. doi: 10.1007/11685654_10. URL `http://dx.doi.org/10.1007/11685654_10`.

[147] L. R. Ford and D. R. Fulkerson. Maximal Flow through a Network. *Canadian Journal of Mathematics*, 8:399–404, 1956. URL `http://www.rand.org/pubs/papers/P605/`.

[148] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927. URL `http://eudml.org/doc/211191`.

[149] M. Stoer and F. Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, July 1997. ISSN 0004-5411. doi: 10.1145/263867.263872. URL `http://doi.acm.org/10.1145/263867.263872`.

[150] S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, Utrecht, May 2000. http://www.library.uu.nl/digiarchief/dip/diss/1895620/inhoud.htm.

[151] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[152] T. Jaakkola. course materials for 6.867 machine learning, fall 2006. Online, 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on 20 Aug 2014.

[153] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: Applications to community discovery. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 737–746, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557101. URL `http://doi.acm.org/10.1145/1557019.1557101`.

[154] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks, 1995.*, volume 4, pages 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968.

[155] X. Li. Niching without niching parameters: Particle swarm optimization using a ring topology. *Evolutionary Computation, IEEE Transactions on*, 14(1):150–169, Feb 2010. ISSN 1089-778X. doi: 10.1109/TEVC.2009.2026270.

[156] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In *2013 IEEE Congress on Evolutionary Computation (CEC),*, pages 2337–2344, June 2013. doi: 10.1109/CEC.2013.6557848.

[157] W. M. Spears, D. T. Green, and D. F. Spears. Biases in particle swarm optimization. *Int. J. Swarm. Intell. Res.*, 1(2):34–57, April 2010. ISSN 1947-9263. doi: 10.4018/jsir.2010040103. URL `http://dx.doi.org/10.4018/jsir.2010040103`.

[158] P. McLean. *48540 Lecture Notes on Signal and Systems*. University of Technology, Sydney, 2014.

[159] R. Hassan, B. E. Cohanim, and O. L. de Weck. Comparison of particle swarm optimization and the genetic algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number AIAA-2005-1897, Austin, Texas, April 18-21 2005. American Institute of Aeronautics and Astronautics.

[160] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251 – 280, 1990. ISSN 0747-7171. doi: http://dx.doi.org/10.1016/S0747-7171(08)80013-2. URL `http://www.sciencedirect.com/science/article/pii/S0747717108800132`. Computational algebraic complexity editorial.

[161] F. L. Gall. Powers of tensors and fast matrix multiplication. *CoRR*, abs/1401.7714, 2014.

[162] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998. ISSN 1049-3301. doi: 10.1145/272991.272995. URL `http://doi.acm.org/10.1145/272991.272995`.

[163] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. ISSN 2050-1439. doi: 10.1111/j.1469-1809.1936.tb02137.x. URL `http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x`.

[164] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, Feb. 1978. doi: 10.2307/2683468.

[165] M. Frigge, D. C. Hoaglin, and B. Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, Feb. 1989. doi: 10.2307/2685173.

[166] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *CoRR*, abs/1111.0352, 2011.

[167] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996. ISSN 0163-5808. doi: 10.1145/235968.233324. URL `http://doi.acm.org/10.1145/235968.233324`.

[168] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[169] American Diabetes Association. Diagnosis and classification of diabetes mellitus. *Diabetes Care*, 20(7):1183–1197, Jul 1997.

[170] B. Al-Diri, A. Hunter, and D. Steel. An active contour model for segmenting and measuring retinal vessels. *Medical Imaging, IEEE Transactions on*, 28(9):1488–1497, Sept 2009. ISSN 0278-0062. doi: 10.1109/TMI.2009.2017941.

[171] X. Jiang and D. Mojon. Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):131–137, Jan 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1159954.

[172] M. E. Martinez-Perez, A. D. Hughes, S. A. Thom, A. A. Bharath, and K. H. Parker. Segmentation of blood vessels from red-free and fluorescein retinal images. *Medical Image Analysis*, 11(1):47 – 61, 2007. ISSN 1361-8415. doi: http://dx.doi.org/10.1016/j.media.2006.11.004. URL `http://www.sciencedirect.com/science/article/pii/S1361841506000909`.

[173] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269, Sep 1989. ISSN 0278-0062. doi: 10.1109/42.34715.