
Enhanced Clustering Analysis Pipeline for Performance Analysis of Parallel Applications



Author: Kaveh Mahdavi

Advisor: Prof. Jesús Labarta Mancho

A THESIS SUBMITTED IN FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Doctor per la Universitat Politècnica de Catalunya
Departament d'Arquitectura de Computadors

Barcelona 2022

Abstract

Clustering analysis is widely used to stratify data in the same cluster when they are similar according to the specific metrics. We can use the cluster analysis to group the CPU burst of a parallel application, and the regions on each process in-between communication calls or calls to the parallel runtime. The resulting clusters obtained are the different computational trends or phases that appear in the application. These clusters are useful to understand the behavior of the computation part of the application and focus the analyses on those that present performance issues.

Although density-based clustering algorithms are a powerful and efficient tool to summarize this type of information, their traditional user-guided clustering methodology has many shortcomings and deficiencies in dealing with the complexity of data, the diversity of data structures, high-dimensionality of data, and the dramatic increase in the amount of data. Consequently, the majority of DBSCAN-like algorithms have weaknesses to handle high-dimensionality and/or Multi-density data, and they are sensitive to their hyper-parameter configuration. Furthermore, extracting insight from the obtained clusters is an intuitive and manual task.

To mitigate these weaknesses, we have proposed a new unified approach to replace the user-guided clustering with an automated clustering analysis pipeline, called Enhanced Cluster Identification and Interpretation (ECII) pipeline. To build the pipeline, we propose novel techniques including Robust Independent Feature Selection, Feature Space Curvature Map, Organization Component Analysis, and hyper-parameters tuning to feature selection, density homogenization, cluster interpretation, and model selection which are the main components of our machine learning pipeline. This thesis contributes four new techniques to the Machine Learning field with a particular use case in Performance Analytics field.

The first contribution is a novel unsupervised approach for feature selection on noisy data, called Robust Independent Feature Selection (RIFS). Specifically, we choose a feature subset that contains most of the underlying information, using the same criteria as the Independent component analysis. Simultaneously, the noise is separated as an independent component.

The second contribution of the thesis is a parametric multilinear transformation method to homogenize cluster densities while preserving the topological structure of the dataset, called Feature Space Curvature Map (FSCM). We present a new Gravitational Self-organizing Map to model the feature space curvature by plugging

the concepts of gravity and fabric of space into the Self-organizing Map algorithm to mathematically describe the density structure of the data. To homogenize the cluster density, we introduce a novel mapping mechanism to project the data from the non-Euclidean curved space to a new Euclidean flat space.

The third contribution is a novel topological-based method to study potentially complex high-dimensional categorized data by quantifying their shapes and extracting fine-grain insights from them to interpret the clustering result. We introduce our Organization Component Analysis (OCA) method for the automatic arbitrary cluster-shape study without an assumption about the data distribution.

Finally, to tune the DBSCAN hyper-parameters, we propose a new tuning mechanism by combining techniques from machine learning and optimization domains, and we embed it in the ECII pipeline.

Using this cluster analysis pipeline with the CPU burst data of a parallel application, we provide the developer/analyst with a high-quality SPMD computation structure detection with the added value that reflects the fine grain of the computation regions.

Acknowledgements

I feel profound gratitude while expressing my sincere thanks to my supervisor Jesus Labarta for his guidance and support throughout my Ph.D. I am immensely grateful to him for offering critical feedbacks on my ideas, and thorough discussions. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I am thankful to the Barcelona Supercomputing Center for funding my Ph.D. studies without which this work would not have been possible. I am also thankful to Nuria Sirvent for providing an excellent administrative support during my Ph.D.

I would like to thank all the members of the Performance Tool team for making these places delightful and rewarding to work. I feel lucky to be a part of these excellent stimulating intellectual environments. My sincere thanks also goes to Judit Gimenes.

I am thankful to my wife for reading an earlier draft of thesis and for upgrading the language. I express my profound gratitude to her support and compassionate behaviour which has always been a continuous source of comfort and strength for me.

Contents

List of Figures	viii
List of Abbreviations	xiii
I Introduction and Related Work	1
1 Introduction	2
1.1 Motivation	2
1.2 Objective	5
1.3 Contributions	7
1.4 Thesis Organization	9
1.5 Publication	10
2 Performance Analysis Field	11
2.1 Description	12
2.2 The Performance Data	13
2.3 Dispatched Data	14
2.3.1 Application Profile	14
2.3.2 Event Traces	15
2.4 Performance Analysis Tools	16
2.4.1 Profile based tools	17
2.4.2 Trace-file based tools	20
2.4.3 Performance Analytics	25
2.5 Cluster Analytics	29
3 Introduction to Clustering Analysis Pipeline and Components	33
3.1 Cluster Analysis	34
3.1.1 Centroid-based clustering	34
3.1.2 Hierarchical clustering	36
3.1.3 Density-based clustering	37
3.1.4 Evaluation Metrics	40
3.2 Machine Learning Pipeline	43
3.2.1 Machine Learning Pipelines	43

3.2.2	Hyperparameters Tuning	45
3.2.3	Unsupervised Machine Learning Pipeline	46
3.3	Feature Selection	47
3.3.1	Foundations of Feature Selection	48
3.3.2	Feature selection methods	49
3.4	Feature Transformation	54
3.4.1	Individual Feature Transformation	55
3.4.2	Multi-Feature Transformation	57
3.5	Clustering Result Interpretation	60
 II New Techniques to Enhance the Clustering Analysis		62
 4 New Unsupervised Feature Selection Technique for Noisy Data		63
4.1	Introduction	63
4.2	Related Works	65
4.3	Background	66
4.3.1	Independent Component Analysis (ICA)	66
4.3.2	Oblique Rotation.	67
4.4	RIFS Algorithm Description	68
4.4.1	Computational Complexity Analysis	69
4.5	Empirical Study	70
4.5.1	Parameter Selection	70
4.5.2	Data Sets	70
4.5.3	Study of Unsupervised Cases	71
4.5.4	Study of Supervised Cases	75
 5 New Method to Homogenize the Density		78
5.1	Introduction	78
5.2	Related work	80
5.3	The problem of multi-density	82
5.4	Background	83
5.4.1	Self-organizing map	84
5.4.2	Multilinear Transformation	85
5.5	Feature Space Curvature Map	87
5.5.1	Feature Space Curvature Modeling	88
5.5.2	Curvature Map	90
5.6	Application of FSCM in the Real Data	93
5.6.1	Datasets	93
5.6.2	Evaluation Metric	94
5.6.3	Experiment Setup	95
5.6.4	Clustering Results	97
5.6.5	Complexity Analysis	98

6	New Method for Extracting Insights from the Shape of Cluster	101
6.1	Introduction	101
6.2	Related work	104
6.3	Background and Notation	105
6.4	Organization Component Analysis	107
6.5	Application of OCA in the Real Data	110
6.5.1	Parameter Selection	111
6.5.2	Evaluation Quality of the Map	111
6.5.3	Identifying Spatial Patterns of Wilderness Sub-area.	111
6.5.4	Diagnosing Performance Bottleneck in HPC Applications.	115
7	Enhanced Cluster Identification and Interpretation Pipeline	120
7.1	Background and Motivation	120
7.1.1	The Limitation of the DBSCAN	122
7.2	ECII Pipeline Architecture	125
7.2.1	Hyperparameter Optimization	126
7.2.2	Evaluation Metrics	128
7.2.3	Components Interaction	129
7.3	Practical Uses to Application Analysis	133
7.3.1	GROMACS	133
7.3.2	Computation Bursts and Enhanced Cluster Analysis	133
7.3.3	Application Analyses	134
8	Conclusion	149
8.1	Feature Selection for Noisy Data	149
8.2	Extracting Insights from the Shape of Cluster	150
8.3	Homogenizing the Clusters Density	150
8.4	Enhanced Cluster Identification and Interpretation Pipeline	151
8.5	Future Work	152
	Bibliography	154

List of Figures

2.1	Example of the flat profile produced by gprof.	17
2.2	Example of the call profile produced by gprof.	18
2.3	Example of the call profile produced by hpcviewer.	19
2.4	Example of the scatter plot of the Completed Instructions counter produced by HPCTOOLKIT's hpcviewer, from HPCTOOLKIT manual [40].	19
2.5	Scalasca Cube workflow.	20
2.6	TAU's ParaProf 3D profile window	20
2.7	TAU's ParaProf Callsite profiling and tracing.	21
2.8	TAU's ParaProf 3D communication window.	21
2.9	Master timeline of VAMPIR trace analyzer.	22
2.10	A Kiviat chart mode of VAMPIR trace analyzer.	22
2.11	Paraver time-line window.	23
2.12	hpcviewer example showing Flat a view.	23
2.13	Vampir summary windows.	24
2.14	Paraver statistics windows.	25
2.15	EXPERT overall architecture. Figure adapted from [46].	27
2.16	Example of PerfExplorer cluster analysis.	30
2.17	Example of PerfExplorer correlation analysis.	30
2.18	A Paraver time-line and profile showing information related to a cluster analysis	31
3.1	kmeanflow	35
3.2	Example of hierarchical clustering: clusters are consecutively merged with the most nearby clusters. The length of the vertical dendogram-lines reflect the nearness.	36
3.3	DBSCANflow	38
3.4	KNN plot for choosing epsilon in DBSCAN algorithm.	39

List of Figures

3.5	Kneedle algorithm for online knee detection. (a) depicts the smoothed and normalized data, with dashed bars indicating the perpendicular distance from $y = x$ with the maximum distance indicated. (b) shows the same data, but this time the dashed bars are rotated $\theta = 45$ degrees. The magnitude of these bars correspond to the difference values used in Kneedle. (c) shows the plot of these difference values and the corresponding threshold values. The knee is found at $x = 0.22$ and is detected $eps = 0.55$, from [83].	40
3.6	A standard Machine Learning pipeline, from [91].	44
3.7	Generic Filter base Feature Selection Algorithm.	51
3.8	Generic Wrapper feature selectors algorithm.	53
4.1	Gaussian noisy versions of the image sample from COIL20 data set with different σ^2 . From left to right σ^2 is: 0, 0.1, 0.4 and 0.7.	64
4.2	Clustering performance vs. the number of selected features on YALE.	72
4.3	Clustering performance vs. the number of selected features on Isolet.	73
4.4	Clustering performance vs. the number of selected features on USPS.	73
4.5	Clustering performance vs. the number of selected features on COIL20.	73
4.6	The noise level vs. the number of selected feature that is needed to achieve the 95% of clustering performance with all features.	74
4.7	Classification accuracy vs. the number of selected features on YALE.	75
4.8	Classification accuracy vs. the number of selected features on Isolet.	76
4.9	Classification accuracy vs. the number of selected features on USPS.	76
4.10	Classification accuracy vs. the number of selected features on COIL20.	76
4.11	The noise level vs. the number of selected feature that is needed to achieve the 95% of classification accuracy with all features.	77
5.1	The stars (distinguished by colors) bend space-time (grid), and the gravitational force among the stars (black lines) can be described by space-time curvature.	79
5.2	A bilinear transformation enables us to represent the arbitrary shaped quadrilateral as a rectangle.	86
5.3	Application of FSCM on a multi-density 2D dataset Synt10 containing ten clusters. (a) A scatter plot of clusters with varied densities. The legend shows the $size/\mu(x^{(1)}, y^{(2)})/\sigma$ per cluster, the colors represent the data original labeling and the red lines draw the initial FSF. (b) shows the FSC model that is computed with our FSCM method. Note that the red lines show the deformation of the FSF. (c) scatter plots the data (a) projected by applying our transformation through model (b). As a result, the diversity of the clusters' density scaled appropriately to achieve a better density-based clustering performance.	87

List of Figures

5.4	Data point transformation between a bent FSC (a) and a regular FSF (b) based on the Multilinear Mapping in \mathbb{R}^2	92
5.5	Comparison of 3D original histogram (a) of dataset Syn10, previously shown in Fig.5.3a, and the homogenized density (c) by our FSCM method, subsequently their DBSCAN clustering results (b) and (d). Without applying FSCM, DBSCAN ends up merging the three blobs in the top right into a single one in order to be able to identify some cluster point in the sparse blob on the left. FSCM as a preprocessing step to DBSCAN allows to better identify the overall structure of the data.	94
5.6	Application of FSCM-DBSCAN on the Wifi dataset and the original datasets in \mathbb{R}^2	99
5.7	Application of FSCM-DBSCAN on the Breast dataset and the original datasets in \mathbb{R}^2	100
6.1	Clusters of diverse shapes in \mathbb{R}^2	102
6.2	Application of OCA to spatial pattern indicator from the Covertypes dataset cluster C3, which is a part of our empirical study. The Horizontal_distance_to_hydrology and Elevation are selected active features. (a) shows the locations of the data points (green) and the weighted graph that is embedded in the SOM topology space. (b) The blue points are representing the SOM neurons, while the red and blue lines are major and minor axis, respectively. The rest of the items are described in the section IV in details.	107
6.3	Comanche Peak Wilderness Area, visualizations of the clustering result (DBSCAN $eps = 0.15$, $minPts = 30$).	111
6.4	Application of OCA to Covertypes dataset cluster C1; (a) shows the data points and the SOM embedded graph.(b) The red and blue dashed arrows represent first and second Organization Component, respectively.	112
6.5	Application of PCA to Comanche Peak Wilderness detected sub-areas C1. The larger the value of the contribution is, the more the feature contributes to the components.	114
6.6	Performance data extracted from STREAM benchmark execution ($OMP_threads_number = 40$, $Loop_size = 9M$) , visualizations of the clustering result (DBSCAN $eps = 0.015$, $minPts = 6$).	116
6.7	Application of OCA to STREAM dataset cluster C1; (a) shows the locations of the data points and the weight vectors.(b) The blue points are representing the SOM neurons, while the red and blue dashed arrows represent first and second Organization component, respectively.	116
6.8	STREAM benchmark, the plots are shown the contour plots of the mean value of IPC and INS, versus the log(loop size) and the $OMP_threads_number$.	118

List of Figures

6.9	STREAM benchmark, the plots present the contour plots of illustrative feature's Directional Sequence Similarity with the major organization component, versus the log(loop size) and the OMP_threads_number. The red dashed lines show the threshold of three hierarchical levels of caches (32kB, 1MB and 33MB) receptively.	119
7.1	The result of a survey of how data scientists spend their time [245] . .	121
7.2	Example of dataset of varies density across the feature space.	125
7.3	ECII pipeline architecture.	129
7.4	Time-lines of different performance hardware counter metrics of GROMACS application executed with 64 tasks.	135
7.5	The existing approach to tune the DBSCAN hyper-parameters [73]. (a) The sorted 16-dist graph was obtained from the GROMACS application. The blue dots represent the distance to the 16th nearest neighbor for each point in the dataset. We use it to compute the different eps values. (b) The search space to select the optimal eps value.	136
7.6	Cluster analysis of GROMACS application using DBSCAN clustering algorithm over Completed Instructions and IPC. Due to the use of a restrictive eps (lower bound) value, the detected structure is noisy. . .	136
7.7	A second cluster analysis of GROMACS application using the DBSCAN cluster algorithm over Completed Instructions and IPC. Selecting a higher value of eps by Grid search produces a coarser grain detection, showing an SPMD structure.	137
7.8	Our ECII approach to tune the DBSCAN hyper-parameters over the Completed Instruction and IPC. (a) The sorted 4-dist and 64-dist graph was obtained from the GROMACS application. The blue and red curve represent the distance to the 4th and 64th nearest neighbor for each point in the dataset. We use it to compute the different configuration of the eps and minPts values. (b) The hyper-parameters search space to select the optimal eps value which is a 2D gird.	138
7.9	The result of our ECII approach to tune the DBSCAN hyper-parameters over the Completed Instruction and IPC. The $X - axis$, $Y - axis$, and $Z - axis$ represent the eps, minPts, and Average Silhouette Width values respectively. The red dashed cycle indicates the optimal hyper-parameter configuration.	139
7.10	Computation structure detection of GROMACS application by applying DBSCAN cluster algorithm with and without the hyper-parameters tuning over the Completed Instruction and IPC.	140

List of Figures

7.11	Clustering scatter plot of GROMACS application using clustering algorithm over Completed Instructions and IPC (a), and its view over Completed Instructions and L1_Rat (b). The blue and black boxes highlight clouds of points that can be divided into isolated groups, and the red box highlight the clouds of points that can be detected as an isolated group by changing the feature subset. Note that the coloring is identical.	142
7.12	Clustering scatter plot of GROMACS application by using the DBSCAN cluster algorithm result over Completed Instructions and L1_Rat with the parameters $minPts = 4$ and $eps = 0.0106$	142
7.13	Computation structure detection of the GROMACS application, using DBSCAN cluster algorithm over Completed Instructions and L1_Rat with the parameters $minPts = 4$ and $eps = 0.0106$	143
7.14	Application of the FSCM on the GROMACS application, and using the DBSCAN cluster algorithm over Completed Instructions and L1_Rat on projected new feature space.	144
7.15	Computation structure detection of GROMACS application by applying DBSCAN ($minPts = 4, eps = 0.0106$) clustering algorithm over Completed Instructions and L1_Rat on the FSCM's homogenized feature space.	145
7.16	Histograms of Locality_L2 and L1_Rat for Cluster 1 in GROMACS application. The patterns of both metrics are almost identical.	148

List of Abbreviations

AI	Artificial Intelligence
BMU	Best Matching Units
CDF	Cumulative Distribution Function
DBI	Davies-Bouldin Index
DNA	Deoxyribonucleic Acid
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ECII	Enhanced Cluster Identification and Interpretation
FPGA	Field-Programmable Gate Array
FSC	Feature Space Curvature
FSCM	Feature Space Curvature Map
FSF	Feature Space Fabric
GIS	Geographic Information System
GPGPU	General-Purpose Graphics Processing Unit
GSOM	Gravitational Self-organizing Map
HPC	High-Performance Computing
HWC	Hardware Counters
ICA	Independent Component Analysis
INS	Completed Instructions
IPC	Instructions per Cycle
kNN	k-Nearest Neighbors
MDCU	Mean Distance to the Closest Unit
ML	Machine Learning
MPI	Message Passing Interface
NMI	Normalized Mutual Information
NNMF	Non-negative Matrix Factorization
OCA	Organization Component Analysis

List of Abbreviations

OPTICS	Ordering points to identify the clustering structure
PCA	Principal Component Analysis
RBF	Radial Basis Function
RIFS	Robust Independent Feature Selection
RRG	Regular Rectangular Grid
SI	Silhouette Index
SOM	Self-organizing Map
SPMD	Single Program, Multiple Data
XAI	Explainable Artificial Intelligence

Part I

Introduction and Related Work

1

Introduction

Contents

1.1	Motivation	2
1.2	Objective	5
1.3	Contributions	7
1.4	Thesis Organization	9
1.5	Publication	10

IN this thesis, we present novel techniques to improve the performance of clustering analysis workflow. In principle, the work was mainly focused on enhancing the clustering analysis, and guiding non-experts to easily build robust solutions for characterizing the performance of parallel applications which usually run on high-performance computing (HPC) systems. It turns out later our research outcomes are generally beneficial in the machine learning domain. Therefore, a particular emphasis of our research is placed on proposing generic machine learning techniques to boost the clustering analysis performance. This chapter presents the motivation for this research and introduces the challenges faced by using traditional Performance Analytics tools. Lastly, this chapter details the list of contributions made by this research to the machine learning literature, as well as the thesis organization.

1.1 Motivation

In the modern world, super-computing is used to approach many of the most important questions in science and technology, including the mathematical modeling for complex

1. Introduction

physical phenomena or designs. The mentioned modeling is such as climate and weather, the evolution of the cosmos, nuclear reactors, new chemical compounds (especially for pharmaceutical purposes), cryptology, and other applications. They can all leverage the massively parallel machines known as supercomputers; however, we need to be careful to use these machines effectively. With the use of multi-core, many-core processors, GPGPUs, deep cache structures, and FPGA accelerators, getting the optimal performance out of such machines becomes increasingly difficult.

On the one hand, the applications that run on these kinds of machines also have many factors that affect their performance. To take advantage of the huge amount of computing power available, the applications must first be parallel. Essentially, a parallel application is an application where parts of its code can be executed at the same time, concurrently, producing partial results that later combine to solve a given problem. In practice, the design and implementation of these parallel applications involve many elements that affect the performance: The sequential algorithms implemented, the distribution of the data used, the communication patterns of the different parallel parts, etc.

High-performance computing (HPC) application developers must take into account this huge number of factors to know and understand how their applications behave on the machine to tune their codes for achieving the "optimum" performance on super-computing platforms. Therefore, the developers are still required to digitize the performance anomalies to design a balanced HPC system. Those anomalies can lead to performance degradation by premature job terminations and wasted compute cycles. Common examples of anomalies include orphan processes leftover from the previous jobs consuming system resources [1], firmware bugs [2], memory leaks [3], CPU throttling for thermal control [4], and resource contention [5, 6]. These anomalies apparent themselves in system logs, hardware counters, or resource usage data. Consequently, they need the performance analysis techniques to mine these data for diagnosing the performance anomalies and their root causes. Although the existing methods can often identify the core grain anomalies, they are still too manual and intuitive. Thus we are interested in exploring what causes a performance loss and where it happens in the code automatically.

In general, the Performance Analysis processes the data and demonstrates various metrics of performance at the level of the program, the functions, the source lines, and the instructions. These metrics can include clock profiling, hardware counter, synchronization delay, memory allocation, MPI tracing, etc. The Performance Analysis tools can also display the raw data in a graphical format as a function of time. The traditional Performance Analysis tool usually can help developers to improve the performance of their applications by producing performance profiles. However, these

1. Introduction

traditional approaches of performance analysis are still an iterative process consisting of observing the behavior of the application to hypothesize the possible bottlenecks that affect its performance and finally turn these hypotheses to improvements in the application restarting the process to validate them. As a result, the difficulty involved in the process of diagnosing and fixing performance bottlenecks since this process is time-consuming and requires a significant amount of guesswork. Furthermore, the static code and profile analyses are often insufficient for identifying the root causes.

On the one hand, to detect performance anomalies and determine the associated root causes, HPC experts typically monitor system behavior by continuously collecting system logs along with hardware performance counters and resource usage data such as available network link bandwidth and CPU utilization. Hundreds of metrics collected from thousands of nodes at frequencies suitable for performance analysis generated during an application execution are huge: up to millions of performance events per second, [7]. As HPC systems grow in size and complexity, it is becoming dramatically impractical to analyze this data manually. Thus, it is essential to have tools that automatically identify the performance anomalies through advanced data analysis techniques.

On the other hand, Data Analytics is the process of systematically applying statistical and machine learning techniques to draw useful information, have conclusions, and inductive inferences from raw data. Performance Analytics is Data Analytics applied to performance analysis data. As presented in the next chapter, there are works included in the different analysis toolkits under the umbrella of this term. In general, many existing performance analysis toolkits offer simplistic manipulations of the performance data. Although First-order statistics such as average or standard deviation are often used to summarize the values of a given performance metric, hiding in some cases interesting facts available from the raw data since they are very limited techniques.

Considering the necessity to summarize the information in a more intelligent way than profiles do, [8] found cluster analysis can be better suited. As defined in [9], clustering is the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait often proximity according to some defined similarity measure. Indeed, it is a classification and not an aggregation of the information. As a result, clustering analysis can identify different latent trends in the data. That is the main characteristic of clustering to overcome the intrinsic weakness of the profiles. Although in [8], they show that density-based clustering techniques are successful and powerful tools to analyze the parallel application performance, the existing density-based clustering algorithms involve lots of arbitrary decisions to provide the best solution including feature selecting, density homogenizing, and noise isolating.

1. Introduction

Furthermore, it turns out that many well-performed clustering results cannot be turned into profound insights easily since the process of making clusters is a generic mathematically oriented process. But, it lacks the intuition and domain knowledge that is often required to interpret and drill down into the algorithmic results. Therefore, there is a need to develop a machine learning pipeline that enables non-experts to conveniently apply the clustering analysis and extracts insight from the result with minimum human supervision.

To consider everything, firstly we need to apply the needed preprocessing steps on a given massive, noisy, and inhomogeneous performance counter dataset to enhance the clustering analysis to optimize its algorithmic performance. Secondly, we have to apply the clustering (e.g. DBSCAN) algorithm by the best-tuned hyper-parameter configuration. Finally, we need to provide a mathematical methodology to extract insight from the obtained clusters, which represents the main motivation of this thesis. Consequently, our main objective is how to improve the clustering analysis process to extract fine-grain insight from enormous amount of high dimensional, noisy, and varied density data, and also how to decipher and present the clustering result mathematically in an understandable way to the non-experts.

1.2 Objective

In general, traditional performance analysis toolkits offer simplistic manipulations of the performance data. First-order statistics such as average or standard deviation are used to summarize the values of a given performance metric, hiding in some cases interesting facts available from the raw data. Thus, we acknowledge that the Cluster Analysis technique is a more powerful and efficient tool to summarize the information according to given criteria that is then a necessity. Due to the complexity of data, the diversity of data structures, high-dimensionality of data, and the dramatic increase in the amount of data, traditional user-guided clustering methodology have many shortcomings and deficiencies in dealing with the problem. Therefore, we motivated to replace the user-guided clustering with an automated clustering analysis pipeline.

The automated pipelines are designed to make the clustering analysis process more accurate, standardized, and faster. However, the adoption of these methods is still limited by the lack of appropriate data preparation, dimensionality reduction, and intuitive result interpretation methods that would allow non-expert users to readily interpret automatically generated clusters. To address these issues, we developed an automated enhanced cluster identification, and interpretation (ECII) pipeline providing robust cluster analysis and actionable insights extracting tools for potentially huge and high-dimensional data and, in particular, performance analysis data.

1. Introduction

In pursuit of this objective, we consider that enhanced clustering analysis techniques are necessary so here we introduce the techniques we propose in this field.

Select most Relevant Feature to Reduce the Dimensionality

The feature subset that we use to build our machine learning models has a significant influence on the algorithmic performance we can achieve. Irrelevant or unimportant features can negatively impact model performance. The Feature Selection is an automatic or manual process to select those features that mostly contribute to our final output in which we are interested in. However, in unsupervised learning scenarios, selecting features is a much harder problem, due to the lack of class labels that would facilitate the search for relevant features. Furthermore, almost all traditional unsupervised feature selection methods are not robust against the noise in samples. For that reason, we need an unsupervised approach for feature selection on noisy data.

Homogenize the Multi-density Clusters

The majority of density-based clustering algorithms can not perform properly when data expose very different densities through the feature space. These algorithms implicitly presume that all clusters almost have the same density so they normally use global parameters. Consequently, they are often biased towards finding dense clusters in front of sparse ones. Therefore, we need to transform the original dataset into the new feature space where the clusters have approximately the same density while all inter-cluster regions become globally low-density.

Interpret the Result of Clustering Mathematically

In general, the process of identifying clusters is a generic mathematically oriented process but lacks the intuition and domain knowledge that is often required to interpret and drill down into the algorithmic results. Although many approaches have been proposed in the literature aiming at interpreting the clustering results, most of these methods are intuitive which makes it difficult to separate the definition of the clusters from the perception of an end-user and even to automatize them. Accordingly, we need a new general method to extract insight from clustering results. The mentioned method should be automated and assumption-free, and it should not be dependent on any particular clustering algorithm.

Tune Hyper-parameters to Optimize the Clustering Model

Among different types of clustering methods, the density-based method has advantages as it does not limit itself to the shapes of clusters. Almost all of the existing density-based clustering algorithms require input parameters that are hard to determine but

1. Introduction

have a significant influence on the clustering result. Hyperparameter tuning is the process of determining the right combination of hyperparameters that maximizes the model performance. Manual hyperparameter tuning approach involves experimenting with different sets of hyperparameters i.e. each trial with a set of hyperparameters will be performed manually by the user. However, manual hyper-parameters tuning is a tedious process since there can be many trials, and keeping track can prove costly and time-consuming. On the other hand, automated hyperparameter tuning utilizes existing algorithms to automate the process. Our objective is to use the leveraging of some popular automated hyperparameter optimization methods to optimize our aforementioned clustering analysis pipeline.

Design and Implement an Enhance Clustering Analysis Pipeline

The two fundamental questions in the clustering analysis process are to find the number of clusters and their compositions. The density-based clustering algorithms, such as DBSCAN [10], exist to answer the prior problem. Although several density-based clustering methods, such as OPTICS [11], proposed for the latter problem, they are facing difficulties in meeting the requirements of data preparation, automation, simplicity, and efficiency. This thesis focuses on developing a simple and efficient density-based clustering pipeline that aims to generate nearly optimal clusters for the given datasets automatically.

1.3 Contributions

As a summary of this chapter, we want to emphasize that the thesis introduces four new techniques in the machine learning field, with the target of improving the clustering analysis. Furthermore, we integrate those techniques as an enhanced cluster analysis pipeline to automatically extract the fine grain insight from the performance counter data. More precisely, we present the new feature selection, density homogenizing, and hyper-parameter tuning techniques to improve the clustering analysis for detecting the computational structure and a novel cluster shape analysis technique to extract insight about the data and, in particular, performance analysis data.

In this work, we present the following original contributions:

Design and Validation of a New Feature Selection Algorithm to Refine the Clustering Quality

We propose a new unsupervised approach for feature selection on noisy data, called Robust Independent Feature Selection (RIFS). Specifically, we try to choose a feature subset that contains most of the underlying information, using the same criteria as

1. Introduction

the Independent component analysis (ICA). Simultaneously, the noise is separated as an independent component. The isolation of representative noise samples is achieved using factor oblique rotation whereas noise identification is performed using factor pattern loading.

Design and Validation of a New Density Homogenization Algorithm to Refine the Clustering Quality

We propose a parametric multilinear transformation method to homogenize cluster densities while preserving the topological structure of the dataset. The transformed clusters have approximately the same density while all inter-cluster regions become globally low-density. In our method, the feature space is locally bent by dense data point concentrations the same way as stars bend the space-time dimensions in the Theory of Relativity. We present a new Gravitational Self-organizing Map to model the feature space curvature by plugging the concepts of gravity and fabric of space into the Self-organizing Map algorithm to mathematically describe the density structure of the data. To homogenize the cluster's density, we introduce a novel mapping mechanism to project the data from a non-Euclidean curved space to a new Euclidean flat space. Specifically, this mechanism transfers the basis vectors instead of the feature vectors to guarantee the continuity of the mapping function and optimize the computation cost of the algorithm. As a result, our method can efficiently and explicitly homogenize the density of any dataset globally to then apply existing clustering algorithms without modification.

Design and Validation of a New Cluster Shape Analysis Technique to Mathematically Interpret the Clustering Result

To fulfill the necessity of automatically extracting insight for the clusters, we propose a novel topological-based method to study potentially complex high-dimensional categorized data by quantifying their shapes and extracting fine-grain insights about them to interpret the clustering result, call Organization Component Analysis. It can mathematically study the arbitrary cluster-shape without an assumption about the data distribution. Our method explores a topology-preserving map of a data cluster manifold to extract the main organizational structure of a cluster by leveraging the self-organizing map technique. To do this, we represent the self-organizing map as a graph. We introduce organization components to geometrically describe the shape of clusters and their endogenous phenomena. Specifically, we propose an innovative way to measure the alignment between two sequences of momentum changes on the geodesic path over the embedded graph to quantify the extent to which the feature is related to a given component. As a result, we can describe variability among stratified data, correlated features in terms of the lower number of organization components.

1. Introduction

Design and Validation of a New Hyper-Parameter Tuning Techniques to Improve the DBSCAN Algorithmic Performance

We propose a new technique to automatically tune the DBSCAN hyper-parameter (eps , $minPts$). We introduce a new approach to define a relatively small and more feasible 2D search space, where each dimension represents the eps and $minPts$, and each point in the search space is a vector with a specific value for each hyper-parameter value. We use the Exhaustive Grid Search and Randomized Parameter Optimization [12] methods to search for optimum hyper-parameter configuration over the search space. To evaluate the goodness of a clustering model, we use the Average Silhouette Width (ASW) [13] as the cluster validation index.

The Introduction of a AutoML Pipeline to Enhance the Clustering Analysis Performance and Demonstration of its Suitability on Computation Structure Detection

To improve the quality of the clustering analysis and focus on producing an automated unsupervised clustering algorithm, we implement an enhanced cluster identification and interpretation (ECII) end-to-end construct pipeline that orchestrates the flow of data into, and output from, the clustering models. It includes raw data input, data preparation, feature selection, tuning the model hyper-parameters, and extracting insight from the output.

1.4 Thesis Organization

The rest of this book has the following chapters. Chapter 2 contains a discussion of the previous work in the parallel performance field including the major work in the Performance Analytics field. Chapter 3 contains a discussion of the previous work in the machine learning including machine learning pipeline and its main components such as data transformation, feature selection, and clustering result interpretation. In Chapter 4, we present our new Robust independent Feature Selection algorithm that can identify the most relevant feature subsets in noisy and unlabeled data, that we use to automatically reduce the dimensionality of the potentially high-dimensional data. Chapter 4 introduces our Feature Space Curvature Map method that we propose to homogenize the cluster density which is important to enhance the density-based clustering algorithm to categorize the varied density data. In Chapter 6, we present a novel topological-based method to extract insight from the clustering result. This method enables us to interpret the result of clustering mathematically by the explanation of internal variability of each cluster. Chapter 7 present an Enhanced Cluster Identification and Interpretation (ECII) pipeline that we propose to conduct

1. Introduction

automatic cluster analyses on the potentially huge and high-dimensional data as it is being integrated. In this chapter, we also demonstrate the usefulness of the ECII pipeline using it to analyze an state-of-the-art application. Chapter 8 offers overall conclusions drawn from our thesis and outlines the research direction for future work based on the ideas and results presented in this thesis.

1.5 Publication

[14] Mahdavi, K., Labarta, J., & Gimenez, J. (2019, November). **Unsupervised Feature Selection for Noisy Data**. In International Conference on Advanced Data Mining and Applications (pp. 79-94). Springer, Cham.

[15] Mahdavi, K., Labarta, J., & Gimenez, J. (2021, July). **Organization Component Analysis: The method for extracting insights from the shape of cluster**. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-10). IEEE.

[16] Mahdavi, K., Labarta, J., & Gimenez, J. (2022, July). **Feature Space Curvature Map: A Method To Homogenize The Density**. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-10). IEEE.

2

Performance Analysis Field

Contents

2.1	Description	12
2.2	The Performance Data	13
2.3	Dispatched Data	14
2.3.1	Application Profile	14
2.3.2	Event Traces	15
2.4	Performance Analysis Tools	16
2.4.1	Profile based tools	17
2.4.2	Trace-file based tools	20
2.4.3	Performance Analytics	25
2.5	Cluster Analytics	29

IN this chapter, we describe several generic and basic concepts about performance analysis that are widely used throughout this document. We start by presenting a taxonomy of the state-of-the-art performance tools and the challenges they face in the analysis of large-scale applications. Next, we describe the Clustering Analysis as a typical Performance Analytic to extract useful insight from a large amount of data.

Parallel performance analysis is a process that starts by extracting the raw performance data that will be later analyzed to determine the application structure. In this context, we want to distinguish three important aspects that compose the whole process: when the analysis is done; the data used to characterize the performance; and finally, the different methods, techniques, and applications analyze this data to draw the conclusions.

2.1 Description

Performance analysis is a multidisciplinary subject that has been adopted in system and software development which involves measurement capabilities, simulation, and system modeling. The importance of performance analysis is threefold:

- There may be performance bugs as there are logical bugs, so if performance is important it should be debugged by measurement.
- When designing new or improved systems or programs, a good understanding of the performance of the base system is needed for avoiding future performance bugs.
- To avoid spending resources on fixing obvious, but minor, inefficiencies rather than searching for the real reasons for the poor performance.

Performance analysis tools are pieces of software that help to measure and deliver comprehensive details of the application behavior on a given system. The details provide several metrics associated with the application structure and the underlying execution. A performance tool helps its users (hereafter, analysts) to understand the unknown behavior of the application. In addition, these tools compare the collected measurements against a theory, model, or even previous executions and ultimately help with fixing performance bugs in the application that they consider worth mending.

When it comes to analyzing the application performance on a system, the analyst may explore the performance from the application or the system point of view. This distinction brings two different types of analyses as one part can be changed while the other remains immutable during the experimentation to achieve the proper comparative analyses. On the one hand, it is possible to consider the software as a fixed part and so explore the processor design space for the purposes of improving forthcoming processor generations by applying architectural simulations. On the other hand, the software developer (or the analyst) may need to tweak the application somehow to take the maximum profit of an immutable system. The scope of this thesis focuses on the latter approach in which developers execute their application on a system and need to adapt the software to the underlying processor.

Concerning the performance tools, different aspects help to classify them. The first classification scheme depends on how data are stored and presented to the user. Performance tools either store time-stamped metrics begetting a time-series stored in a trace-file, or summarize all the measurements with the consequent space savings, but almost lose time-dependent issues. A secondary way to classify performance tools refers to the mechanism used for executing the monitors (or probes): instrumentation

2. Performance Analysis Field

and sampling. Instrumentation refers to the ability to inject monitors to specific application locations. Therefore it provides accurate metrics to these regions of code. On the other hand, sampling takes advantage of mechanisms to periodically invoke monitors so that the results are statistical inferences of the application behavior. While some of the tools expose the captured metrics with minimal or no manipulation, some tools process the metrics by applying additional mechanisms and extract conclusions automatically without major intervention from the analyst.

This chapter covers most of the topics relating to the performance analysis tools. First, we describe the performance data and its acquisition mechanism. Second, there will be an extended summary of a variety of state-of-the-art tools, discussing their design points and presenting the reports they provide to the analyst. The summary of the tools will also include the investigation of many techniques to evaluate metrics automatically to ease the analyst's experience. A section will then follow with a detailed dissertation of the existing Performance Analytics methods. Finally, there is a brief discussion about applied clustering techniques to extract insight from the performance data.

2.2 The Performance Data

To understand the behavior of a parallel application on a given machine, we need to measure different elements that reflect its performance. The most simple and recurrent element measured is the application execution time. Undoubtedly, the execution time is a good indicator of the application performance, and in most cases it is the objective value to minimize.

In the performance analysis scenario, it is also common to use time, but a finer grain, for example measuring the time spent on each application subroutine. To perform these measurements, we can make use of the timing mechanisms provided by the operating system (OS), for example, the programmable interval timers. Furthermore, this example points to a second performance metric: the application code location. Location information is required to relate the metrics to the application source code. This location can be a single position where the measurement is taken. It is accessed via the Program Counter (PC) of the CPU, or the call path, which includes the list of active subroutines in the application. We can also use `libunwind` [17] to determine their source code location.

Performance hardware counters (HWC) values are unique metrics to understand the behavior of the application in a given hardware. Hardware counters are available in almost all modern processors and count micro-architectural events, for example the total cycles elapsed or the number of instructions executed. Hardware vendors provide

2. Performance Analysis Field

the hardware and software interface to access these counters, but the PAPI [18] library, a homogeneous application programming interface to access the counters in most of the hardware, is the most common way to read them. More recently, due to the need to better understand new hardware, we can also find performance hardware counters in other components beyond CPUs, such as the Infiniband network hardware [19] or the Nvidia CUDA GPUs [20]. In both cases, the counters are also available via PAPI.

The metrics regarding the parallel programming model are varied, as well as the mechanisms to access them. For example, in the message-passing applications the ones we analyze in this thesis, it is normal to gather the number of messages sent or received, the size of these messages, etc. In this case, if the application uses MPI, we have available the MPI profiling interface (PMPI) [21] to intercept the MPI calls and extract these values. In some other cases, access to the run-time of the programming models requires more sophisticated mechanisms, as detailed in the next section when talking about instrumentation.

Apart from this list, there are a variety of metrics to evaluate specific performance elements (I/O, power consumption, etc.). It is not the aim of this section to present every performance metric available, but those are commonly available in most performance toolkits.

2.3 Dispatched Data

There is a classical difference of opinion over how to dispatch and store the information whether application profiles or event traces. Consequently, the way the information is stored partially determines the kind of available analyses.

2.3.1 Application Profile

An application profile is a summation of a metric, or a set of metrics, that characterizes an application-level abstraction. A typical example of a flat profile is the number of calls and the time spent in each application subroutine. In the process of summation, the temporal component information regarding when the data was collected is lost.

gprof [22] is the classic tool for profiling sequential codes. This tool relies on compiler-assisted instrumentation to define the bounds of the application subroutines and the POSIX timers and signaling to take the samples. It generates partial call graph profiles, more detailed profiles that express the caller-called relationships across the application subroutines.

OpenSpeedshop [23] and the HPCTOOLKIT [24] extend gprof capabilities focusing on parallel applications. Both can manage the information generated by all

2. Performance Analysis Field

tasks/threads involved in a parallel application, to present a single profile. In addition, they provide the ability to profile not only application subroutines but also basic blocks of code, or even single individual source code lines.

gprof, OpenSpeedshop, and HPCTOOLKIT heavily rely on sampling mechanisms to perform the data acquisition. On the other hand, TAU [25] from the University of Oregon provides a profiling toolkit based on instrumentation mechanisms to enable the user to have control of what is going to be profiled, including MPI run-time accesses and performance hardware counters. TAU also gathers phase profiles, partial profiles extracted at different stages of the application execution. In TAU, the phases are defined by the user and can be understood as the logical steps in the application evolution, for example, the different time steps in a weather forecast simulation. The phase profiles provide an approach to observing the time-varying behavior of an application.

The Scalasca toolkit [26] also offers the collection of regular profiles with metrics from MPI, OpenMP, and hardware counters using instrumentation. In this toolkit ecosystem, we also find an interesting effort gathering time-series call-path profiles [27], call-graph oriented version of the previously mentioned phase profiles.

2.3.2 Event Traces

An event trace is the collection of all information gathered during the application execution, consisting of a log or trace file, of the actions that a parallel application performed. These actions are assumed as time-stamped events, including the entry and exit to the subroutines or to the parallel libraries used. It also includes values of the hardware counters, or the call-path that lead to the point of interest. Typically, the generation of event traces relies on instrumentation packages to define the points of interest, but we can also find sample traces. Event traces offer a highly detailed view of the performance, at the cost of high storage space requirements. They are especially interesting to analyze the time-varying behavior of the application, and lost information when using profiles.

In general, almost all parallel performance analysis toolkits offer the extraction of event traces as the base for further analysis. For example, the BSC Tools package, the toolkit used in this thesis, is based on Paraver trace [28]. This trace is a structured text file whose main characteristic is that it is semantic-free: the contained events are essentially timestamped key/value pairs. A complimentary (optional) file is responsible for linking the semantics to the contained tuples in the trace file.

Extrae is the instrumentation and sampling tools of the BSC Tools package that collects metrics from several parallel and accelerator run-times (such as MPI, OpenMP,

2. Performance Analysis Field

CUDA, OpenCL, OmpSs, among others) that generates Paraver trace-files, although other tools have created their own trace-file translators into Paraver format.

As a part of the Score-P initiative [29], the analysis toolkits Scalasca, Vampir [30], Periscope and TAU decided to adopt the Open Trace Format version 2 (OTF2) [31], the second generation of the Open Trace Format [32]. OTF2 is structured in a collection of multiple binary files accessible via an API. The main concern in the design of this trace is the scalability: the trace format definition includes a series of encoding techniques to reduce its size [33], and the access API uses techniques to reduce the memory footprint.

As mentioned before, there are also some efforts to produce sample traces. As an alternative to the regular traces whose events are associated with instrumentation points, sample traces contain a series of time-stamped samples taken regularly during the execution. The information in each sample mainly includes the call-path, to correlate to application source code and performance metrics, such as the hardware counters. We find an early approach to the generation of sample traces inside the Sun Studio Performance Tools (now Oracle Studio) [34]. Recently, the HPCTOOLKIT and the BSC Tools also added sample traces on their toolkit ecosystems, [35] and [36] respectively. It is interesting to highlight that the addition of samples to the Paraver trace described in [36] did not imply any modification of the trace format.

2.4 Performance Analysis Tools

The performance analysis tools need to answer two major questions that analysts face when studying an application. What are the nature of the performance inefficiencies and where are they located within the application? The performance tools describe the application performance behavior using either summary (profiles) or detailing variations in performance across time (trace-files). The profiles apply first-order and second-order statistics (such as mean, min, max, and variance) to the measurements captured to simplify the metrics provided to the analyst. The tools that use trace files allow expressing the variability of performance issues exposing sequence of metrics as well as multi-modal behavior.

In a postmortem scenario, once the data has been collected, the analysis step starts by exploring the gathered information manually or assisted to detect patterns or trends. The mentioned patterns and trends can reflect anomalies or performance losses in the application behavior and correlate them with the possible causes.

At this point, we distinguish between two different elements of the analysis itself. The first is how the information is presented to the analyst to make the analysis process understandable and manageable. The second is the collection of available

2. Performance Analysis Field

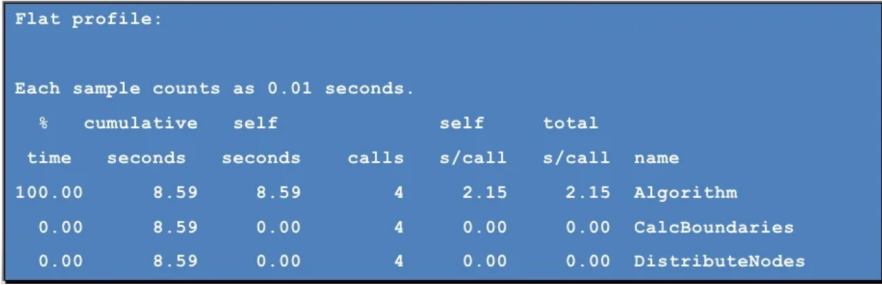
Performance Analytics techniques, i.e. the different techniques developed to automate the processing of the raw data to detect anomalies and correlations.

Data presentation is a key element of the analysis process. The way the information is offered to the analyst defines, to some extent, the possible observations and hypotheses they could make about the performance achieved by the application. Undoubtedly, the way data is emitted imposes a series of restrictions or opportunities on its presentation.

2.4.1 Profile based tools

Application profiles are an explicit example of this data manipulation at once. The different first-order statistics measurements are taken during the execution such as the sum or the average at the selected application-level abstraction (subroutines, application phases, etc.). Using profiles, the exploration of unusual situations can be done easily. For example, sorting the total time spent in the different subroutines will point us to where the hot-spots are. The weak point of the profiles is that the aggregation hides the potential variability across the instances accounted. However, it is a great idea to look at the performance initially as the profiles are a good choice.

In general, profiles are presented as human-readable plain text files. The text is indented to categorize the elements accounted for more easily. The elements are sorted concerning one of the metrics used to focus the analysis. In Figure 2.1 we can see a flat profile produced by gprof. It contains the different metrics calculated for each subroutine. The right-most column is sorted by the percentage of total application time spend and in the left-most column each subroutine is represented. The rest of the metrics includes the exclusive/inclusive times (time inside the subroutine excluding or including the calls it makes) and the number of calls. The call-graph profile in Figure 2.2 contains similar metrics, but the right-most column presents the caller-called relationship using the indentation. These call-graph profiles generated by gprof can also be visualized in an interactive GUI, using a tree representation, with the IBM's Xprofiler [37].



```
Flat profile:

Each sample counts as 0.01 seconds.

%   cumulative   self           self         total
time  seconds     seconds       calls   s/call     s/call  name
100.00    8.59    8.59            4      2.15      2.15  Algorithm
  0.00    8.59    0.00            4      0.00      0.00  CalcBoundaries
  0.00    8.59    0.00            4      0.00      0.00  DistributeNodes
```

Figure 2.1: Example of the flat profile produced by gprof.

2. Performance Analysis Field

```

Call graph
granularity: each sample hit covers 4 byte(s) for 0.48% of 2.08 seconds
index % time    self  children  called  name
                2.08   0.00    1/1    main [2]
[1]  100.0    2.08   0.00    1      Algorithm [1]
-----
                                <spontaneous>
[2]  100.0    0.00   2.08
                2.08   0.00    1/1    Algorithm [1]
                0.00   0.00    1/1    DistributeNodes [4]
                0.00   0.00    1/1    CalcBoundaries [3]
-----
                                0.00   0.00    1/1    main [2]
[3]   0.0    0.00   0.00    1      CalcBoundaries [3]
-----
                                0.00   0.00    1/1    main [2]
[4]   0.0    0.00   0.00    1      DistributeNodes [4]
-----

```

Figure 2.2: Example of the call profile produced by gprof.

In contrast to the simple plain-text document, we find tools such as the `hpcviewer` [38], an evolution of the `HPCVIEW` [39], which is the profile visualization tool of the `HPCTOOLKIT`. The `hpcviewer` is a GUI that offers a clean presentation of the different metrics gathered in the `HPCTOOLKIT` profiles, with the ability to link the metrics with the source code, shown in Figure 2.3, or present the metrics using charts for better comprehension, shown in Figure 2.4.

Scalasca’s `CUBE` [41] visualizer has an original way to visualize and interact with profiles. The main display of this GUI is divided into three parts (see Figure 2.5). The left part shows the different metrics gathered; the central part is the code location; and the right part shows the system location, i.e. the accounting of each metric in the physical hardware. Thanks to the coloring, the user can see the severity of a given metric i.e. low, regular, or high values quickly.

Finally, `ParaProf` [42], the profile analysis tool of the `TAU` package, offers similar functionality to the `hpcviewer`, in the areas of source code correlation and the generation of the different chart types. In Figure 2.6, we can see an example of a 3D profile presentation of the most time-consuming application subroutines, including the MPI primitive calls. In Figure 2.7, we can see another example of profiling and tracing for `MPI_Ranks`. Figure 2.8 shows a `ParaProf` communication matrix among the senders and receivers where the colors represent the message size. In addition, `ParaProf` is also capable of comparing profiles obtained from different executions of the same application. For this purpose, `TAU` uses their `PerfDMF` [43], a framework to manage profiles from different executions using a database that eases the comparison of profiles.

2. Performance Analysis Field

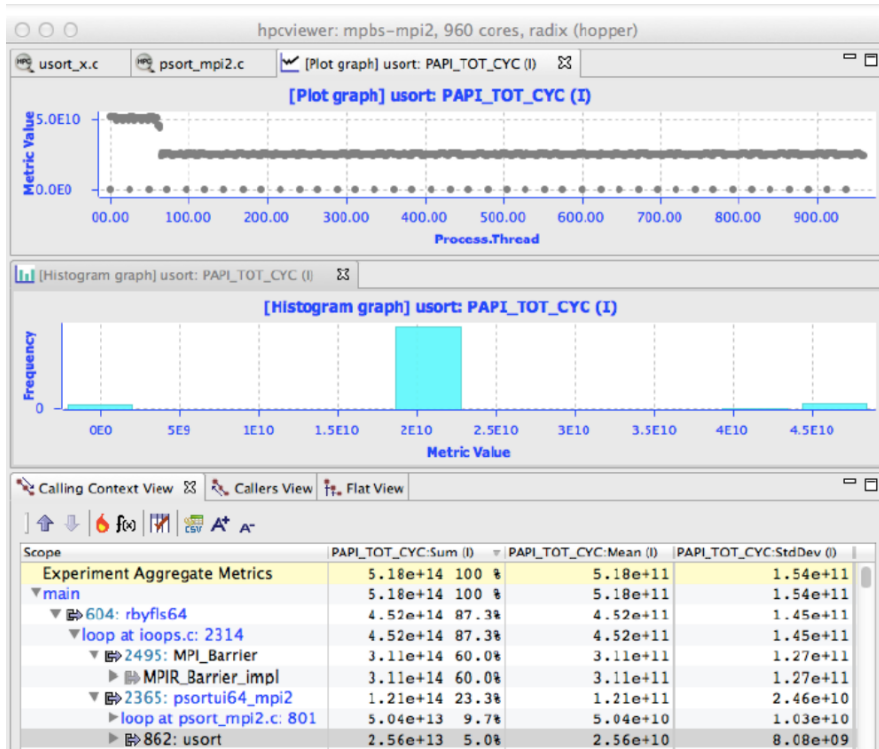


Figure 2.3: Example of the call profile produced by hpcviewer.

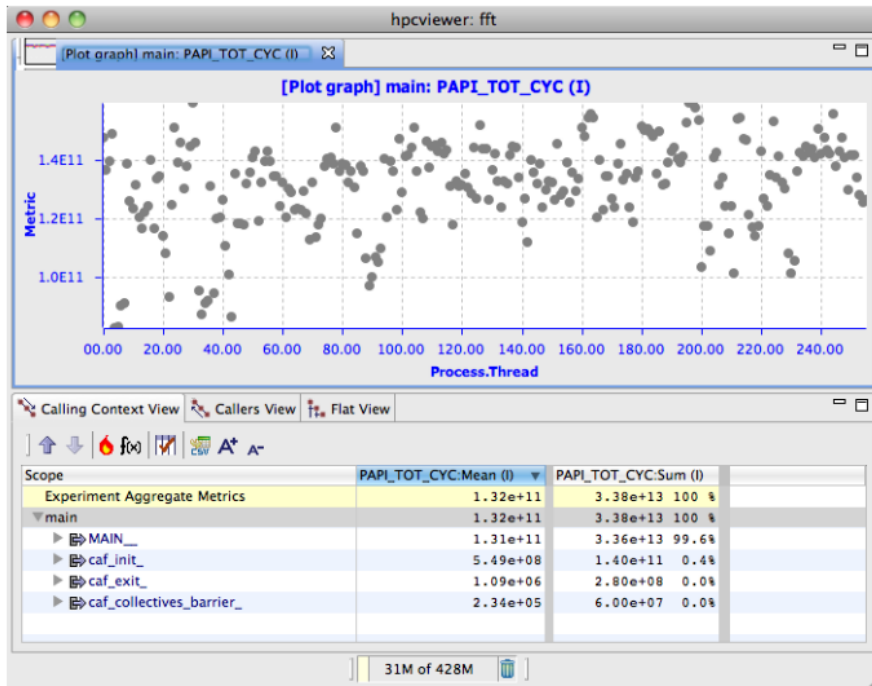


Figure 2.4: Example of the scatter plot of the Completed Instructions counter produced by HPCTOOLKIT's hpcviewer, from HPCTOOLKIT manual [40].

2. Performance Analysis Field

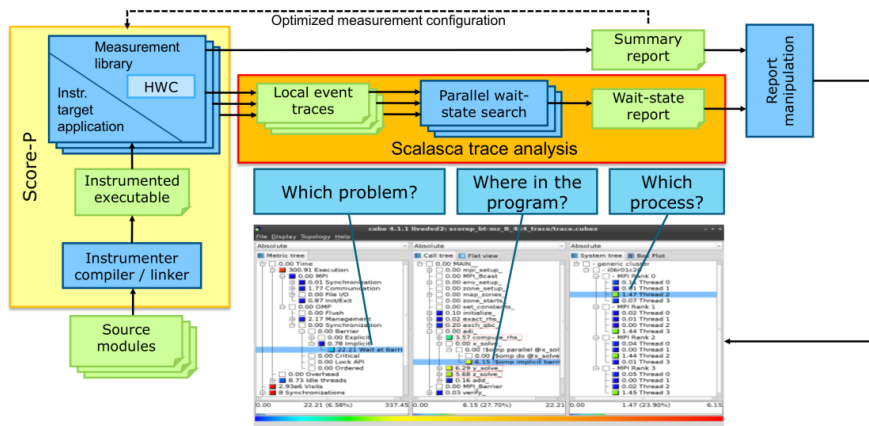


Figure 2.5: Scalasca Cube workflow.

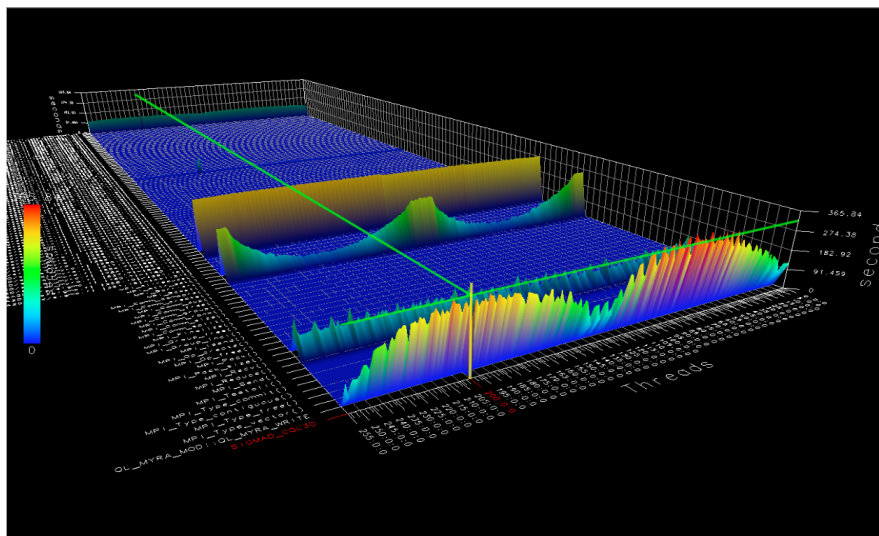


Figure 2.6: TAU's ParaProf 3D profile window

2.4.2 Trace-file based tools

As opposed to the profiles, event traces have greater potential to detect elements at a finer granularity, as they keep track of all the actions performed by the analyzed application, including their spatial and temporal distribution of them. On the other hand, traces require more effort both in the manipulation and the exploration to detect the anomalies and correlate them to the root causes.

To aid the analyst, there are available several interactive GUIs such as Vampir [30], Paraver, or the hpctraceviewer. In terms of exploration, all these tools provide a unique visualization which is impossible to obtain when using the profiles. A timeline is a representation where the available information in the trace is organized in a bi-dimensional plot applications abstractions such as tasks or threads versus time, and colored according to a metric.

2. Performance Analysis Field

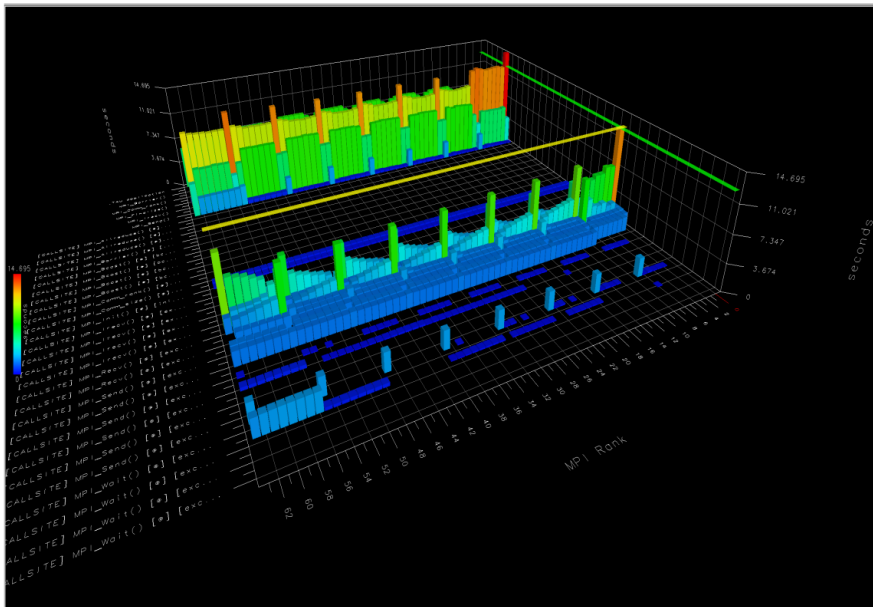


Figure 2.7: TAU's ParaProf Callsite profiling and tracing.

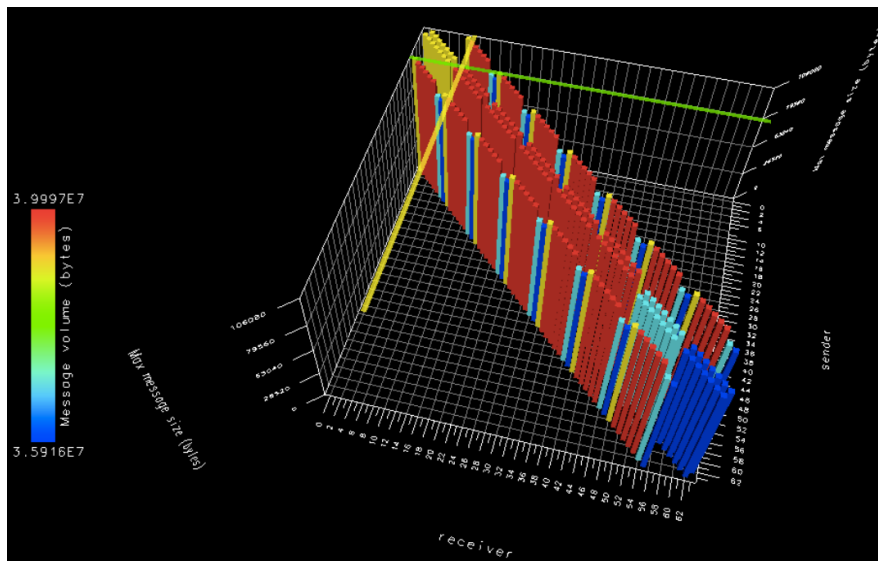


Figure 2.8: TAU's ParaProf 3D communication window.

Figure 2.9 contains an example of the Vampir master timeline, presenting an OTF trace. In this case, the Y-axis represents the processes of a message-passing application. The coloring indicates the subroutine executed. We can also observe black lines that represent the point-to-point messages passed between the different processes. In the top left part of the window, we can see a general view of the whole trace. Being the main time-line is just a zoom of the central part. Furthermore, we can use the Kiviat mode to show the accumulated time per function, see 2.10.

In Figure 2.11 we can see an example of the Paraver timeline. In this example, the metric selected was the Instructions per Cycle (IPC), derived from the Completed

2. Performance Analysis Field

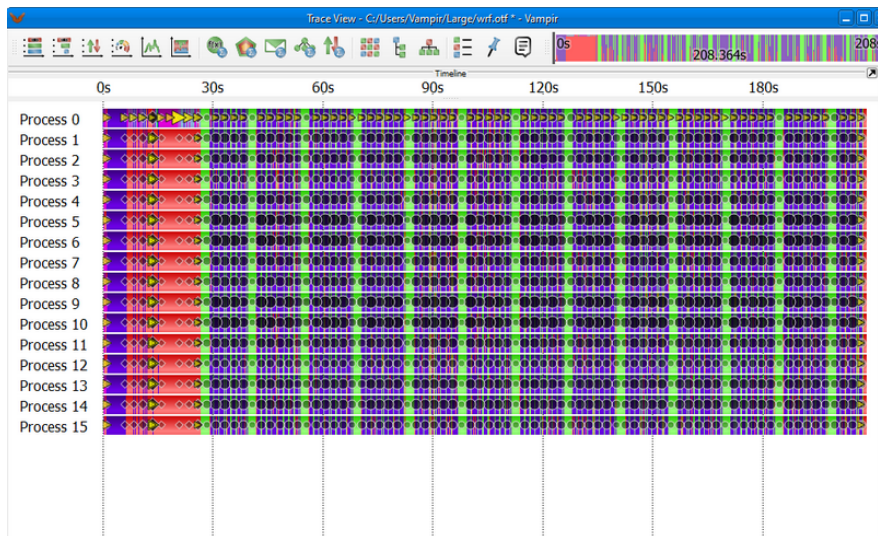


Figure 2.9: Master timeline of VAMPIR trace analyzer.

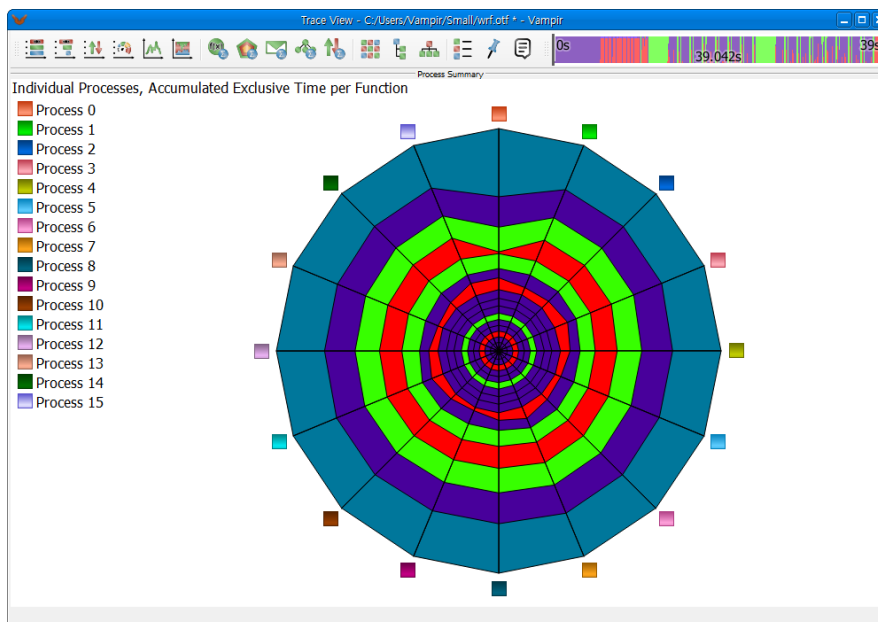


Figure 2.10: A Kiviat chart mode of VAMPIR trace analyzer.

Instructions and Total Cycles hardware counters present on the trace. The coloring is a gradient from light green (low IPC) to dark blue (high IPC). Paraver traces also contain point-to-point message information. These are depicted as yellow lines in the timeline.

Finally, Figure 2.12 contains a window of the hpctraceviewer. In this case, since the presented sample trace does not have information regarding the point-to-point communications, the timeline does not include communication lines. On the other hand, the bottom part of the window presents the depth of the call path obtained on each sample of the trace.

Even though the time-line representation is useful to explore the time distribution

2. Performance Analysis Field

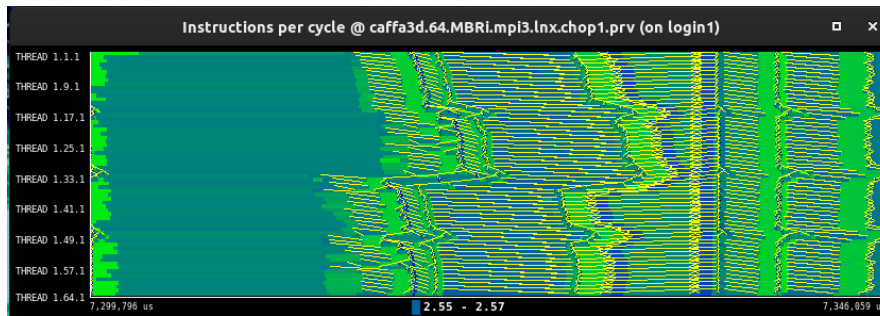


Figure 2.11: Paraver time-line window.

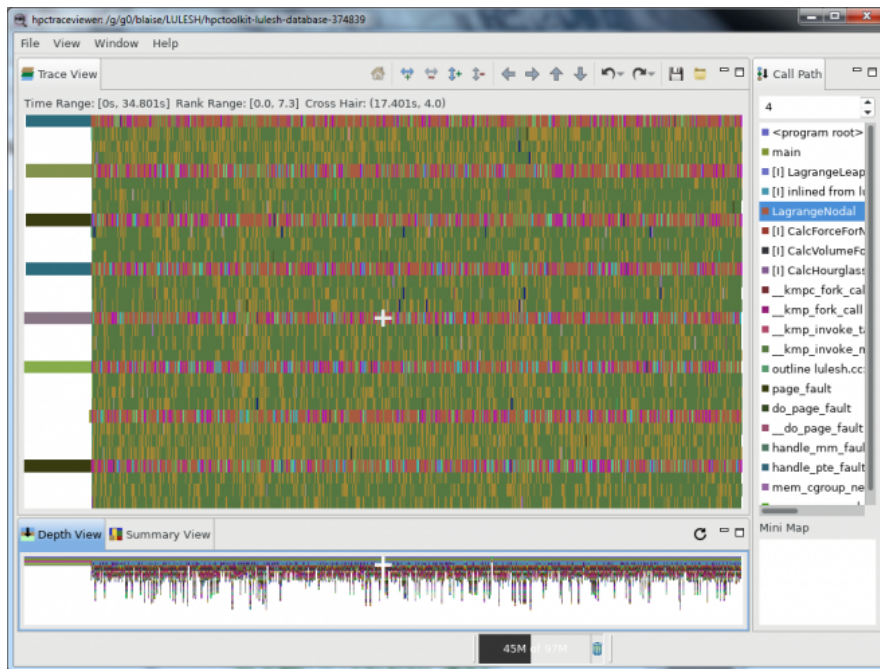
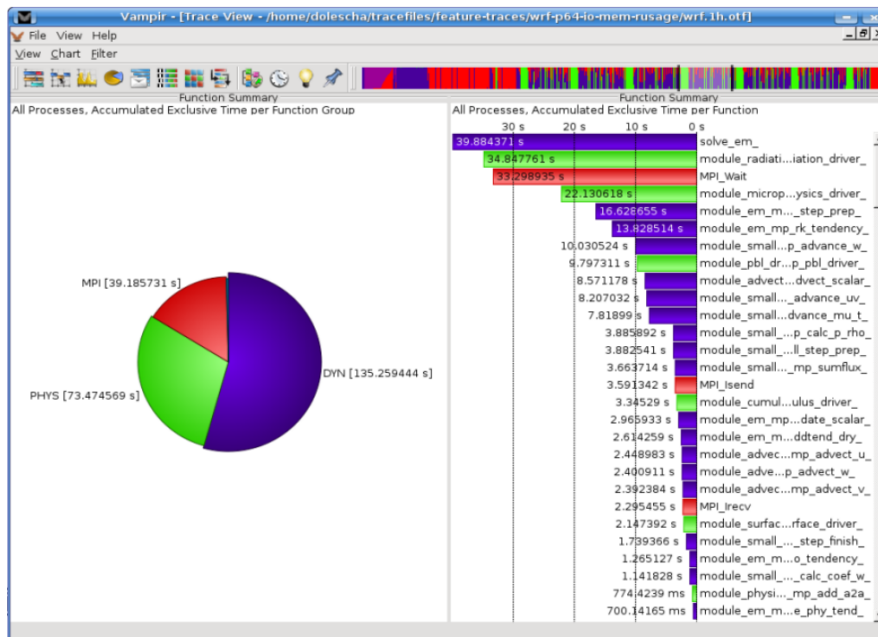


Figure 2.12: hpcviewer example showing Flat a view.

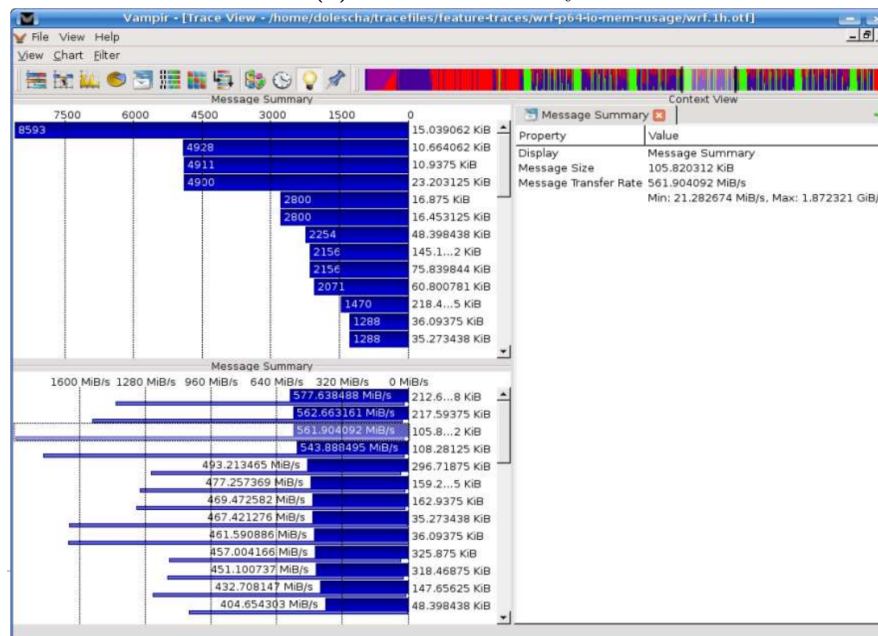
of raw (or derived) metrics at different levels of granularity using zooming, there are limitations to this representation. First, current screen resolutions limit the amount of data that can be presented by the human eyes' bandwidth. Second, there is a biological limit to distinguish the differences of the color hue. These limitations imply that the representation requires a big effort in processing the input data to render how each pixel of the bitmap is filled. For example, a single pixel of the screen may represent more than one object or a wide range of time, so an algorithm is required to decide how to depict the actual value. For example, non-linear renderings of the data ranges are used to clarify the representation of the presented different values.

To overcome these constraints, the trace analyzers, mainly Vampir and Paraver, also include a set of features to manipulate the available information in the event trace. In the case of Vampir, it offers a battery of predefined summaries that include the generation of flat-profiles, Figure 2.13a, and also message size summary, see Figure

2. Performance Analysis Field



(a) Function summary.



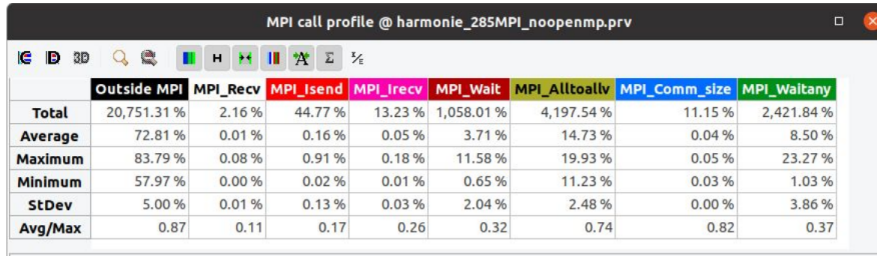
(b) Message Summary

Figure 2.13: Vampir summary windows.

2.13b. Paraver also includes a profiling view but extends the regular summation with the ability to freely combine different metrics available, to detect the possible correlations across them. For example, Figure 2.14a shows MPI statistics, to know the load balance, parallel efficiency, etc. For instance, the load balancing is of great importance when utilizing multiple processors as efficient as possible. Adding more processors creates a noticeable amount of synchronization overhead. Therefore, it is only beneficial if there is enough work present to keep all processors busy at the

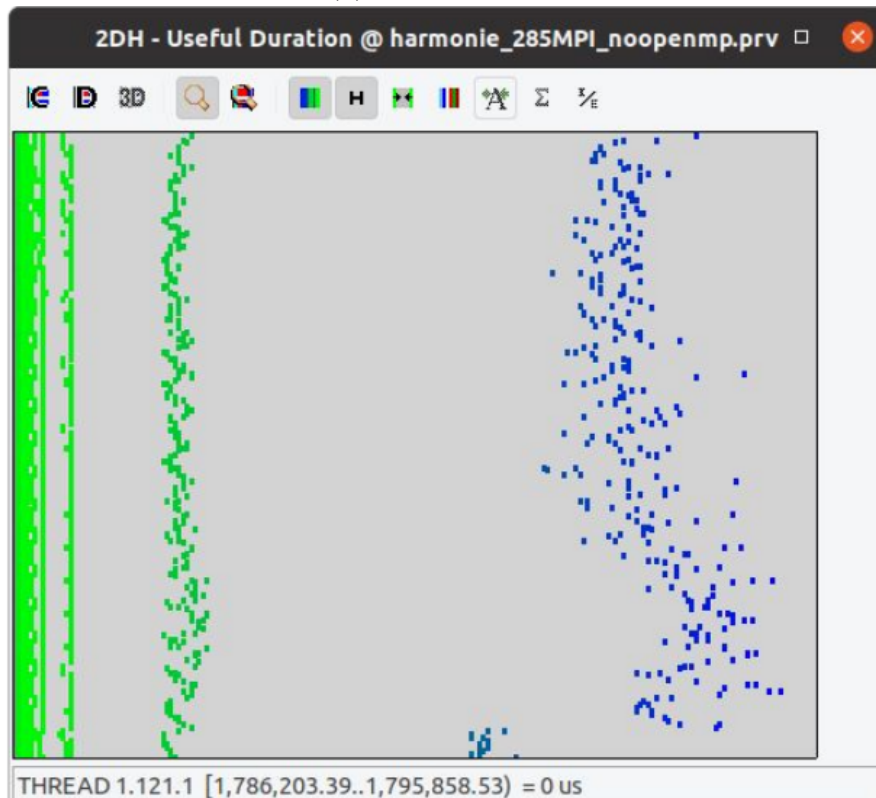
2. Performance Analysis Field

same time. In Figure 2.14b, we can see a complex histogram of the duration of the computational chunks to study the load balance. Even though these two tools differ in the features to combine performance metrics, both can compute a wide range of statistics, similar to profiles: sums, averages, standard deviations, etc.



	Outside MPI	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Alltoallv	MPI_Comm_size	MPI_Waitany
Total	20,751.31 %	2.16 %	44.77 %	13.23 %	1,058.01 %	4,197.54 %	11.15 %	2,421.84 %
Average	72.81 %	0.01 %	0.16 %	0.05 %	3.71 %	14.73 %	0.04 %	8.50 %
Maximum	83.79 %	0.08 %	0.91 %	0.18 %	11.58 %	19.93 %	0.05 %	23.27 %
Minimum	57.97 %	0.00 %	0.02 %	0.01 %	0.65 %	11.23 %	0.03 %	1.03 %
StDev	5.00 %	0.01 %	0.13 %	0.03 %	2.04 %	2.48 %	0.00 %	3.86 %
Avg/Max	0.87	0.11	0.17	0.26	0.32	0.74	0.82	0.37

(a) MPI call profile.



(b) Metrics combination histogram of useful duration.

Figure 2.14: Paraver statistics windows.

2.4.3 Performance Analytics

As a summary of the previous section, we can conclude that the application profiles provide a coarse-grain knowledge of the application but do not require big expertise when looking for possible hot-spots. So, when going into detail, event traces are necessary, at the cost of requiring big expertise to manipulate the available information

2. Performance Analysis Field

to detect the performance problems and correlate them to their causes. The goal of Performance Analytics is to join the best of both worlds. In other words, to be able to get the deep insight provided by event traces without requiring the expertise to manipulate the huge amount of information.

The Performance Analytics are the techniques that translate the expert knowledge into algorithms or methodologies to be able automatically extract the most valuable information about applications performance. In general, these techniques rely on the search patterns or trends that characterize the performance losses. The search can be done at different granularities and at the different levels of abstraction obtaining the different analysis results.

2.4.3.1 Inference Systems

A methodology widely exploited is the use of a rule-based inference system to detect the well-known problems of parallel applications, [44]. An example of a rule can be: “*IF* the time spent by a task waiting to send a message is bigger than x because the partner task has not executed the receive operation *THEN* late receiver problem detected”. Usually, the rules include a severity value, indicating the importance of the detected problem in the performance of the whole application.

As a part of the Scalasca toolkit, EXPERT [45–47] is a sequential postmortem rule-based system to identify the wait states in message-passing applications. Figure 2.15 presents the EXPERT overall inference systems architecture. In this work, the set of rules of known wait states, i.e. when the application is wasting time without advancing in the resolution of the problem, is defined using the *EARL* script language and applied to the performance metrics stored in OTF traces. As a result, EXPERT generates profiles that can be visualized in CUBE. In the profiles, EXPERT accounts for the occurrences of each problem in the knowledge-based on each of the subroutines. Further research on the EXPERT system includes a parallel implementation [48] and an extension of the rule-based in [49] to track the root cause of the wait states detected, i.e. those regions or points in the application that later provoke wait states when application communicates.

KappaPI [50] and KappaPI 2 [51], are two versions of the same rule-based tool with the difference that KappaPI includes the set of rules hard-coded, while KappaPI 2 brings the user the possibility of writing his own rules, using the APART Specification Language (ASL) [52]. This language has been adopted in some of the rule-based due to its power to easily express performance problems. An interesting feature of both tools is the recommendation system, which shows possible solutions to the detected problems.

The SCALEA toolkit [53] also provides similar features to the Scalasca toolkit, with the support of a multi-experiment environment based on database storage as

2. Performance Analysis Field

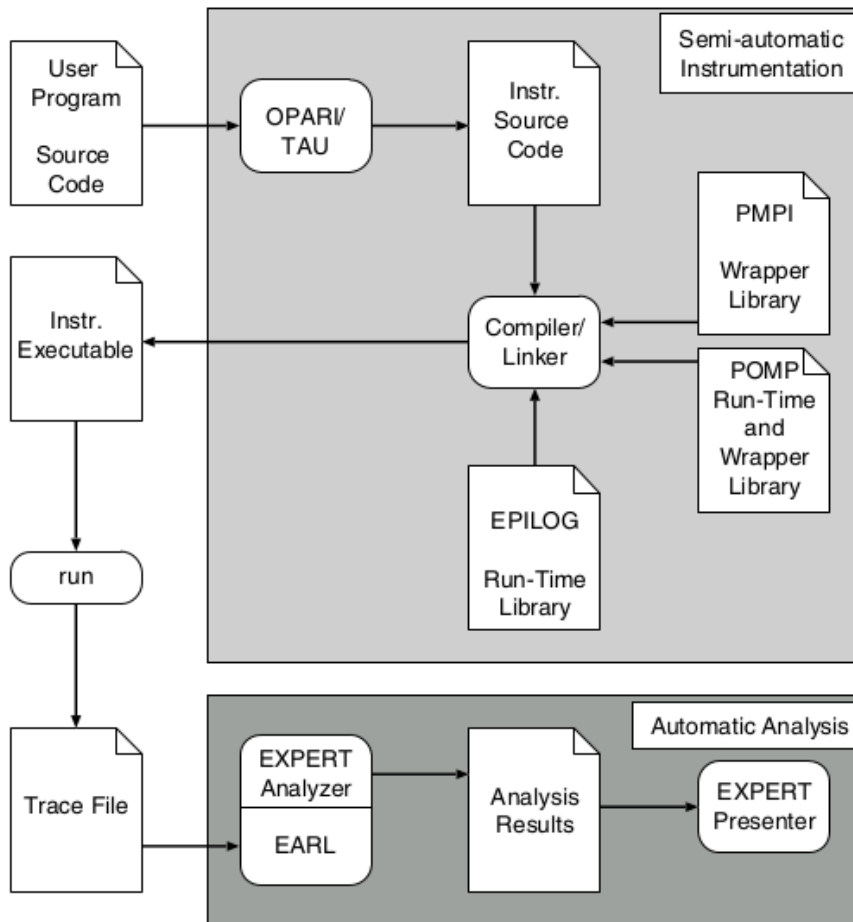


Figure 2.15: EXPERT overall architecture. Figure adapted from [46].

PerfDB. Aksum [54, 55] makes use of the data extraction mechanisms provided by SCALEA to implement its rule-based system. In this case, the ruleset used has to be specified in JavaPSL, a Java version of the previously mentioned ASL.

While all works mentioned above do postmortem analysis, there are also two major works of rule-based online analysis. The first one is the Performance Consultant Module, part of the Paradyn [56, 57]. Periscope [58, 59] is essentially a Paradyn clone, with the possibility of define custom rules using ASL.

The Paradyn Parallel Performance Tools project [56] opened the direction of intelligent selection of performance data leveraging a technique called dynamic instrumentation [60].

Paradyn is a performance tool which relies on run-time performance analysis, rather than recording a complete trace of the whole execution. The user can select which performance metrics they wish to view. Additionally, it also provides a bottleneck search algorithm named the Performance Consultant (PC), that automatically searches for a set of known performance problems. This algorithm uses the W3 model to

2. Performance Analysis Field

guide the bottleneck search, which attempts to answer why, where and when the application is performing poorly. This is an iterative process of formulating hypotheses and refining them based on the performance data being collected. For example, if a routine performing synchronization operations is taking up a large amount of time, the hypothesis of `ExcessiveSyncTime` will be formulated, and instrumentation will be dynamically inserted into that routine in order to collect more specific data that helps to find the specific causes or to reject the initial hypothesis. The PC periodically gathers performance data from every process and decisions about how the bottleneck search progress is taken centrally.

The instrumentation can be achieved through rewriting tools like Dyninst [61]. DynInst is an instrumentation library that allows us to modify the application by injecting code at specific locations. With the instrumentation capabilities of DynInst, we can instrument different parallel programming runtimes as OpenMP, CUDA accelerated applications, and MPI applications.

Periscope [59] is an online profile that automatically searches for bottlenecks based on previous optimization experts' knowledge. Periscope offers several bottleneck analyses depending on the application type and the architecture where the application runs including load imbalance, excessive time MPI time due to several reasons, spent time in OpenMP exclusion regions and so. The results of Periscope are integrated into Eclipse through a plugin that helps to associate the obtained metrics with the source code in a single environment

2.4.3.2 Pattern Recognition

Other interesting methods to add intelligence to the analysis rely on the fact that parallel applications usually have a repetitive structure to detect patterns in application phases. For example, Casas et. al. [62] applies signal processing techniques to Paraver traces to detect coarse-grain iterations of the application algorithms. The resulting tool produces a summary of the application defined by the patterns found. It also generates partial traces containing only the repetitive patterns, orders of magnitude smaller than the original, that can be used for later analysis in the detail.

Freitag et. al. [63] presents the Dynamic Periodicity Detector (DPD) an online analyzer of OpenMP parallel function patterns used in the BSC Tools performance data extraction library, `Extrae`. It uses the stream of OpenMP parallel function identifiers to detect repetitive sequences on the subroutine calls, to generate smaller Paraver traces where the repetitive sequences had been compressed.

A third work project in pattern recognition can be found in [64]. In this case, the recognition is done at visualization time in a Vampir module. This approach tries to solve the problem of visualizing large amounts of data, showing the repetitive

2. Performance Analysis Field

patterns on the trace as “boxes” on a timeline view. The pattern detection is based on Compressed Complete Call Graphs (cCCG) [65] that optimizes the application of call tree representation to save space.

2.5 Cluster Analytics

In this dissertation, we place our emphasis on enhancing the cluster analysis techniques. These techniques whose target is to classify elements into groups have also been exploited previously in the performance analysis area, but with different objectives.

Nickolayev et al. [66], based on [67], proposes the application of the K-means clustering algorithm in an online analysis to determine the similarities among processors involved in parallel application execution. The authors use coarse-grain granularity metrics such as processor idle or running times to describe the behavior of each processor. The work was developed as a part of the Pablo Performance Environment, with the target of reducing the event traces generated. Instead of flushing the performance data of all processors, the output trace just includes the metrics of a representative processor per cluster detected, reducing the output data up to 4.5 times.

In [68], Ahn and Vetter develop a deep statistical analysis of event traces containing processor performance hardware counters that characterize application subroutines. In this work, hierarchical clustering and K-means clustering are used to determine the similarity across the processes, MPI tasks, and OpenMP threads, at the level of subroutines. The authors demonstrate the utility of clustering to automatically distinguish master-slave patterns of the processes and also application algorithm structural patterns such as the organization of the processes depending on the problem decomposition. Furthermore, the authors use other multivariate statistical methods, Principal Components Analysis (PCA) [69] and Factor Analysis [70], to highlight the high correlations between some of the performance counters. These two techniques are useful to select those metrics that provide more information, reducing the dimensionality of the collected data.

The framework PerfExplorer [71], developed as a part of the TAU toolkit, offers similar features to [68] with major emphasis on describing the detected clusters, see Figure 2.16. In PerfExplorer, K-means and hierarchical clustering, PCA, Factor Analysis, and Correlation Analysis are applied to profiling information stored in a PerfDMF database, which includes a wide variety of metrics, from high-level idle or running times to low-level processor hardware counters. For example, Figure 2.17 shows a correlation analysis result. As in [68], the clustering algorithms are used to find the parallel processes, both MPI tasks, and OpenMP threads, behave similarly. The major contribution of this work is the correlation of the groups found

2. Performance Analysis Field

with the profiling information available, which provides the analyst with a clear understanding of the behavior of the clusters. Furthermore, PerfExplorer implements a rich GUI to navigate through this information.

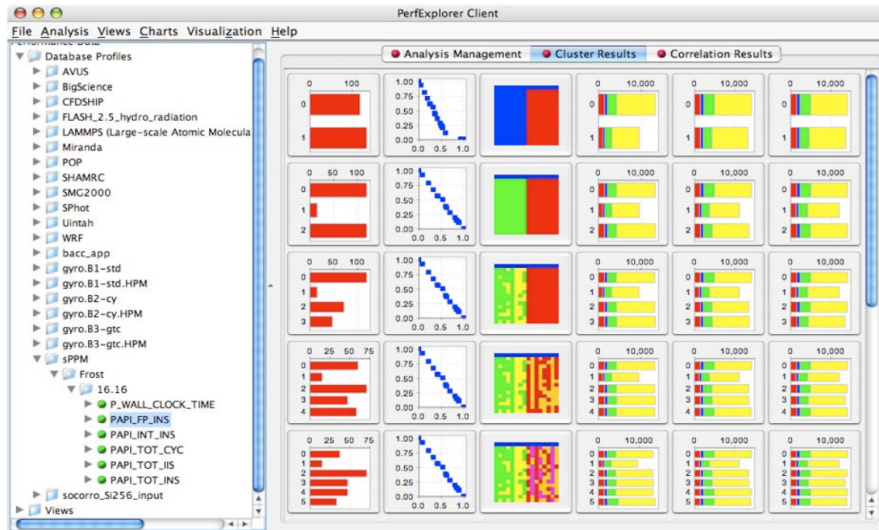


Figure 2.16: Example of PerfExplorer cluster analysis.

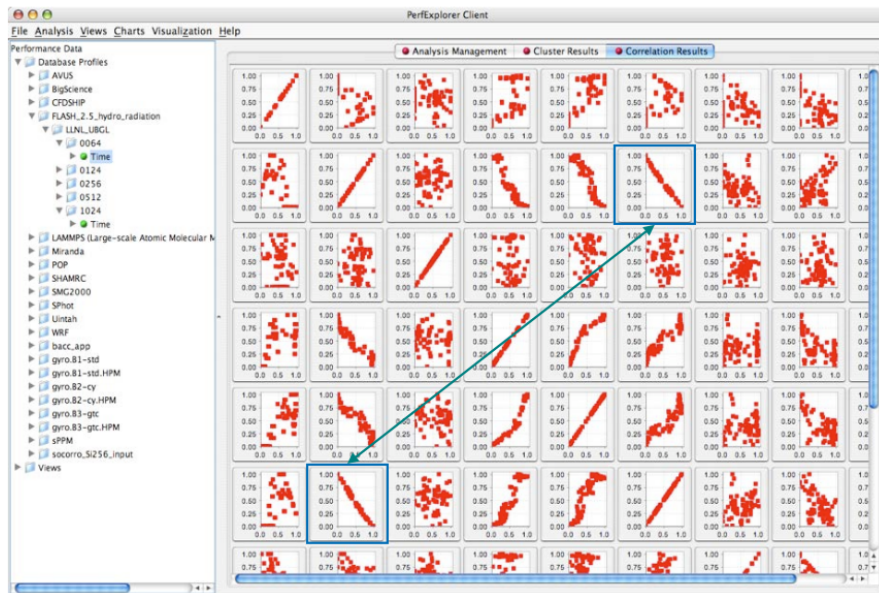


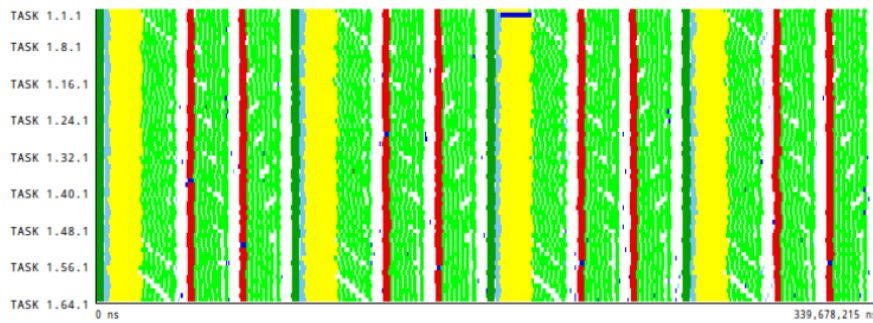
Figure 2.17: Example of PerfExplorer correlation analysis.

In [72], they found a different use of the cluster. The goal of the work is also to reduce the information, in this case, the instructions needed for an accurate micro-architectural simulation, but the characterization tries to find similar application phases. The execution phases are regions of 100 million instructions. The used metrics to characterize them are basic block vectors, uni-dimensional vectors that

2. Performance Analysis Field

account for how many times each of the basic blocks of the application has been executed in the given region. The authors demonstrate that K-means can correctly distinguish the application phases and that simulate a set of representatives, not just the centroid, to provide high-quality simulations.

In [8], they demonstrate the suitability of density-based cluster algorithms, specifically DBSCAN, when characterizing the computation structure of parallel applications. They show that this kind of algorithm surpasses the clustering algorithms based on K-Means when grouping performance hardware counters (HWC) data, and work more effectively than hierarchical clustering algorithms because they do not require user interaction. As an example, Figure 2.18 presents a Paraver time-line and profile showing information related to a cluster analysis.



(a) Time-line distribution of discovered clusters.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
TASK 1.1.1	103,371,000 ns	53,163,000 ns	16,951,000 ns	11,573,000 ns
	101,279,000 ns	40,256,000 ns	19,191,000 ns	11,871,000 ns
	108,464,000 ns	54,570,000 ns	19,586,000 ns	11,895,000 ns
	108,587,000 ns	54,543,000 ns	19,852,000 ns	11,898,000 ns
	106,993,000 ns	54,539,000 ns	19,700,000 ns	11,818,000 ns
	108,652,000 ns	54,529,000 ns	19,631,000 ns	11,850,000 ns
TASK 1.8.1	95,878,000 ns	54,660,000 ns	19,683,000 ns	11,892,000 ns
	101,991,000 ns	53,945,000 ns	17,353,000 ns	11,490,000 ns
	96,053,000 ns	53,805,000 ns	20,579,000 ns	12,015,000 ns
	103,801,000 ns	53,615,000 ns	21,393,000 ns	12,348,000 ns
	109,804,000 ns	54,624,000 ns	21,986,000 ns	12,328,000 ns
	110,764,000 ns	54,577,000 ns	21,927,000 ns	12,368,000 ns
TASK 1.16.1	111,561,000 ns	54,528,000 ns	21,877,000 ns	12,269,000 ns
	100,226,000 ns	54,607,000 ns	22,040,000 ns	12,320,000 ns
	96,208,000 ns	53,757,000 ns	20,143,000 ns	12,266,000 ns
	100,976,000 ns	53,729,000 ns	19,193,000 ns	11,916,000 ns
	95,966,000 ns	53,682,000 ns	20,646,000 ns	11,920,000 ns
	104,536,000 ns	53,613,000 ns	21,416,000 ns	12,313,000 ns
Cluster 1	110,702,000 ns	54,448,000 ns	21,805,000 ns	12,220,000 ns
	109,171,000 ns	54,436,000 ns	22,099,000 ns	12,275,000 ns

(b) Duration histogram of the clusters per application task.

Figure 2.18: A Paraver time-line and profile showing information related to a cluster analysis

Subsequently, they propose the Aggregative Cluster Refinement [73] algorithm to improve the quality of the structure detection by focusing on producing a purely unsupervised cluster algorithm. It is the combination of a density-based algorithm

2. Performance Analysis Field

and a hierarchical algorithm that maximizes the quality of a score. Using the Cluster Sequence Score, the Aggregative Cluster Refinement can detect SPMD computation regions in message-passing parallel applications at different levels of granularity.

3

Introduction to Clustering Analysis Pipeline and Components

Contents

3.1 Cluster Analysis	34
3.1.1 Centroid-based clustering	34
3.1.2 Hierarchical clustering	36
3.1.3 Density-based clustering	37
3.1.4 Evaluation Metrics	40
3.2 Machine Learning Pipeline	43
3.2.1 Machine Learning Pipelines	43
3.2.2 Hyperparameters Tuning	45
3.2.3 Unsupervised Machine Learning Pipeline	46
3.3 Feature Selection	47
3.3.1 Foundations of Feature Selection	48
3.3.2 Feature selection methods	49
3.4 Feature Transformation	54
3.4.1 Individual Feature Transformation	55
3.4.2 Multi-Feature Transformation	57
3.5 Clustering Result Interpretation	60

IN this chapter, we describe several generic and basic concepts about machine learning that are widely used in this document. We describe a typical clustering analysis workflow and the common challenges to apply it. Then, we touch on the overall benefits of the Machine Learning Pipeline to automatically build an optimized machine learning model without requiring high expertise to manipulate a large amount of information and introduce the relevant algorithms and techniques for each step

3. Introduction to Clustering Analysis Pipeline and Components

of the model development workflow. Finally, we explain how the main goals of this thesis align with the current state-of-the-art and put the presented work in context.

Applied machine learning is typically focused on finding a single model that performs well or best on a given dataset. Effective use of the model will require appropriate preparation of the input data including feature selection, feature extraction, transformations, etc, and hyperparameter tuning of the model.

Collectively, the linear sequence of steps required to prepare the data, tune the model, and transform the predictions is called the modeling pipeline. Modern machine learning approaches build the pipeline to codify and automate the workflow which produces a machine learning model.

In this chapter, we present a brief introduction to clustering analysis algorithms and the needed machine learning techniques which are required to fully understand the technical contributions of this thesis.

3.1 Cluster Analysis

Cluster analysis consists of assigning a set of objects into groups as the clusters. The objects assigned to the same cluster are similar in terms of a distance or dissimilarity metric. In this thesis, these objects are d -dimensional points where each dimension is a performance metric. The used distance measure is the Euclidean Distance.

This clustering task can be implemented in different ways obtaining a wide set of distinct clustering algorithms. Surveys by P. Berkhin [74] and by Xu and Wunsch [75] review a high number of these algorithms, but in this section, we focus on algorithm families most commonly used in HPC tools.

3.1.1 Centroid-based clustering

In the centroid-based clustering algorithms family, the resulting clusters are represented by a central vector, which can be part of the data set (a *medoid*) or not (a *centroid*). The rest of the objects in the data set are assigned to the nearest cluster center.

K-means [76] algorithm is the classic example of the centroid-based clustering algorithm. It divides the whole set of data points in k clusters C_j . Each cluster is represented by the mean value (or weighted average) c_j , the centroid. The sum of discrepancies between a point and its centroid is used as an objective function in the iterative optimization schema. The usual distance to minimize is the Euclidean distance.

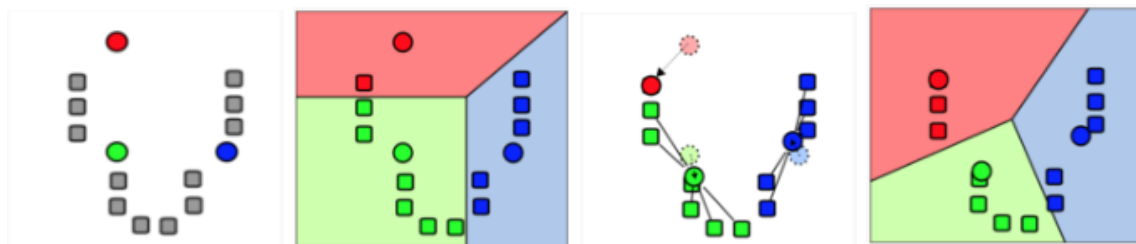
The algorithm requires the user to supply the desired number of clusters k . The optimization schema is composed of three simple steps:

3. Introduction to Clustering Analysis Pipeline and Components

- Compute the centroids. In the first iteration, they can be computed randomly or by several other methods. In further iterations, the centroids are the mean value of the points assigned to each cluster.
- Compute the distances of each point to all the centroids.
- Assign the points to the closest centroid, to minimize the discrepancies.

When each iteration is finished, the algorithm checks if there has been any change in the points assigned to each cluster. If the clusters have not changed, the algorithm finishes.

Figure 3.1 illustrates this process. The first plot 3.1a is the election of initial centroids, the round shaped points. Plots 3.1b and 3.1c are the iterative nucleus of the algorithm, where points are assigned to their nearest centroid and then the centroids are updated. The algorithm converged when objects assigned to each cluster do not change in two consecutive iterations, plot 3.1d.



(a) Step 1: generation of initial centroids). (b) Step 2: assign points to closest centroid. (c) Step 3: recompute the centroids. (d) End: Convergence state.

Figure 3.1: A graphical example of K-means algorithm using $k = 3$. After generating the initial centroids in Step 1, points of the data set are assigned to the closest one on Step 2, and centroids are recomputed on Step 3. Steps 2 and 3 are executed until the algorithm converges.¹

The selection of k is one of the major drawbacks of K-means/medoids algorithms as it is not always possible to guess how many different clusters are presented in the data set. The algorithm X-means [77] tackles this problem by applying the Bayesian Information Criteria (BIC) that evaluates the quality of clusters in terms of how well they represent Gaussian distributions. X-means iteratively executes K-means increasing the value of k , verifying on each iteration if the resulting clusters increase the BIC score.

The equivalent algorithm but using the medoids which is called K-medoids being Partition Around Medoids (PAM) [78] the reference implementation.

¹https://en.wikipedia.org/wiki/K-means_clustering

3. Introduction to Clustering Analysis Pipeline and Components

By far, centroid-based algorithms are the most widely used clustering algorithms. The major advantage of these algorithms is the easy implementation and also its fast execution. Unfortunately, these algorithms lack some important aspects: they are not robust against outliers and are assumed a hyper-spherical structure of the data distribution.

3.1.2 Hierarchical clustering

The target of hierarchical [79], also called connectivity-based clustering, is the construction of a hierarchy of individuals in a data set. This hierarchy is represented as a dendrogram, a tree where the leaves are the individuals and the root represents the whole data set. Intermediate nodes represent the groups of two or more individuals whose height concerning the leaves expresses the value of a linkage metric required to conform such a group. This linkage metric is based on a dissimilarity function and can be computed in different ways: single linkage uses the minimum value of the dissimilarity metric between two points/groups; or average linkage, the average of the distance; full linkage uses the maximum value of the dissimilarity metric.

Figure 3.2 contains an example of a small data set, the resulting dendrogram obtained with a single linkage of the Euclidean distance, plot 3.2a. Points A and B, and C and E, have the same distance, so they merge at the same height in the dendrogram. Next, the group of A and B merges with C at slightly higher height. The rest of the dendrogram expresses a merge of the group formed D, E, F. Finally the whole data set is merged at the top level.

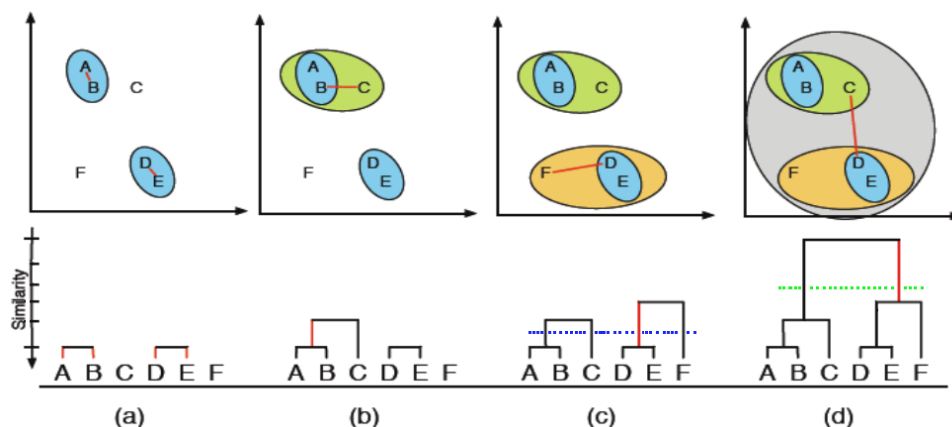


Figure 3.2: Example of hierarchical clustering: clusters are consecutively merged with the most nearby clusters. The length of the vertical dendrogram-lines reflect the nearness.

The strategy to build the dendrogram results in two types of hierarchical clustering: aggregative, a bottom-up approach, merging the individuals/groups from leaves to

3. Introduction to Clustering Analysis Pipeline and Components

root; or divisive, a top-down approach, separating the individuals/groups from the root to the leaves.

To obtain a data partition in a hierarchical clustering, it is required to perform a horizontal cut at some height in dendrogram. In Figure 3.2 there is an example of two different partitions: plot 3.2c shows a partition obtained by cutting the dendrogram at the height marked by blue dotted-line; analogously, plot 3.2d corresponds to the clusters obtained by cutting at the heights marked by green dotted-line. We can see that a cut close to the leaves produces a big number of clusters more compact. In contrast, a cut close to the root of the dendrogram produce less clusters with more variability.

Opposite to the centroid-based clustering, where the algorithms generate Gaussian clusters around a centroid/medoid, hierarchical clustering does not assume the underlying model of the data. In other words, the construction hierarchy of the individuals according to the dissimilarity metric is orthogonal to how the individuals are distributed. On the other hand, deciding which level of the dendrogram is expressing the most valuable division of the data to perform the cut is a hard task. This problem worsens when dealing with large data sets due to difficulty to depict and analyze the dendrogram.

3.1.3 Density-based clustering

Density-based clustering algorithms are partition algorithms, as K-means, but they share some features with connectivity-based clustering. The aim of density-based clustering is to group points which are linked by a particular connectivity property and their density is big enough to be considered as a real cluster. These algorithms are widely used in the image recognition area.

DBSCAN [80], Density-Based Spatial Clustering of Applications with Noise, is a classic example of density-based algorithm. The inputs of DBSCAN are two parameters, the radius Epsilon (*eps* or ϵ) and minimum number of points (*minPts*). This algorithm is based on two basic definitions:

1. A point p is directly density-reachable from a point q if their distance is less or equal to *eps*, and the *eps*-neighbourhood of q (neighbours in a distance less or equal than *eps*) is greater or equal to *minPts*. This relation is not symmetric.
2. Two points p and q are density reachable if there is a sequence p_1, \dots, p_n , where $p = p_1$, $q = p_n$, and each $p_i + 1$ is directly density-reachable from p_i .
3. Two points p and q are density connected if there is a point o such that both p and q are density reachable from o .

3. Introduction to Clustering Analysis Pipeline and Components

The resulting clusters obtained by DBSCAN are those subsets C_i of the data where all points in the subset are mutually density connected. The points that do not fall in a cluster are considered noise. Note that the definitions lead to two types of points inside a cluster: *core* points, inner points in a cluster that serves to fulfill the density connectivity across all points in the cluster, and *border* points, in the edges of the cluster, from where there are no directly density-reachable points available. These definitions are clarified with Figure 3.3: red points are the core points of the cluster formed by red and blue points; blue points are border points; the colored circumferences are the range of the ϵ -neighbourhood; red points are a noise point that does not belong to the cluster.

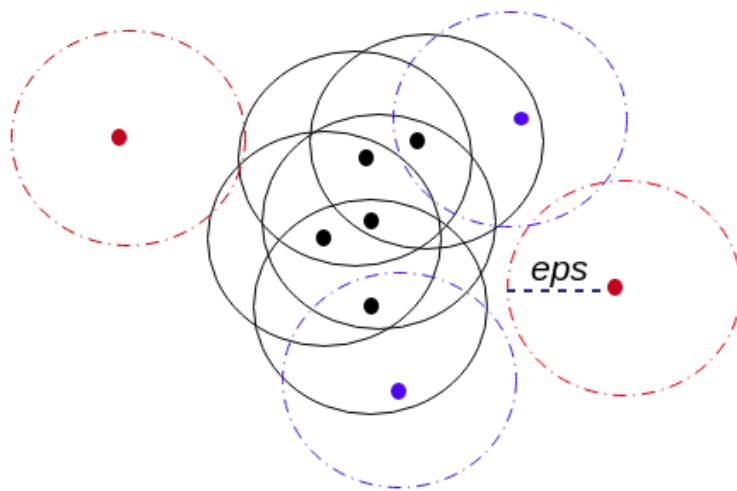


Figure 3.3: A single DBSCAN cluster with Core, Border, and Noise Points. Blue points are border points. The black points are core points, and the red point are a noise point.²

As hierarchical clustering, density-based clustering algorithms do not do any considerations about the data structure or model. In addition, there are robust against outliers and noise. An important disadvantage is the lack of interpretability of the resulting clusters.

3.1.3.1 Estimate the Epsilon

As previously mentioned, we must provide a value for epsilon which defines the maximum distance between two points. In [81] they describe an approach for determining the appropriate value for eps. They find a suitable value for epsilon by calculating the average distance to the k nearest neighbors for each data point of the dataset, sorting and plotting the results. Then they look where the change is most pronounced (think of your knee angle) and select that distance as epsilon.

²<https://www.mdpi.com/2071-1050/11/17/4718>

3. Introduction to Clustering Analysis Pipeline and Components

To find the knee of the curve, first define the line connecting the first and last points of the curve, see the dashed black line in Figure 3.4a. The ordinate of the point on the sorted k-dist graph, and the furthest point from the line defines epsilon, see the green arrow in Figure 3.4a .

In [82] they use *MinNumPoints* and *MaxNumPoints* to set a range of k-values for which epsilon is calculated. Then they use the averaged *eps* value as the overall estimated *eps*. The points below this threshold belong to a cluster, while points above this value are identified as noise by DBSCAN. Figure 3.4a shows the k-NN search curves and the estimated epsilon, and Figure 3.4a shows the clusters which are identified by averaged estimated *eps* value.

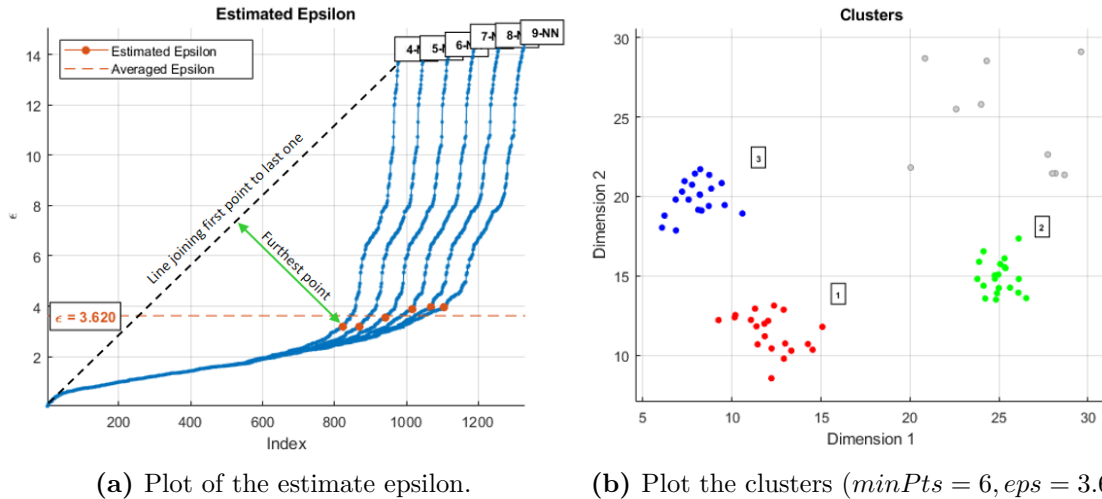


Figure 3.4: KNN plot for choosing epsilon in DBSCAN algorithm.

The Kneedle [83] algorithm can identify the point with the greatest absolute distance from its neighbors by applying more sophisticated schemes than aforementioned techniques. It is based on the notion that the point of maximum curvature in a dataset (the knee) is approximately the point in a curve that are local maxima. If the curve is rotated θ degrees clockwise about (x_{min}, y_{min}) through the line formed by the points (x_{min}, y_{min}) and (x_{max}, y_{max}) [83].

Figure 3.5 depicts how Kneedle algorithm works for data points drawn from the curve $y = -1/x + 5$ where $x \in [0, 1]$. Note that we assume that the curves under consideration have negative concavity. For curves with consistently positive concavity (e.g., forming “elbows” rather than knees), such as k nearest neighbors graph, it is trivial to invert the graph by replacing each y_i with $y_{max} - y_i$ and x_i with $x_{max} - x_i$.

Although they are semi-automatic approaches for estimating the *eps*, DBSCAN is still sensitive to the selected *minPts* range. The *minPts* value should be set using domain knowledge and familiarity with the data set. There is no automatic

3. Introduction to Clustering Analysis Pipeline and Components

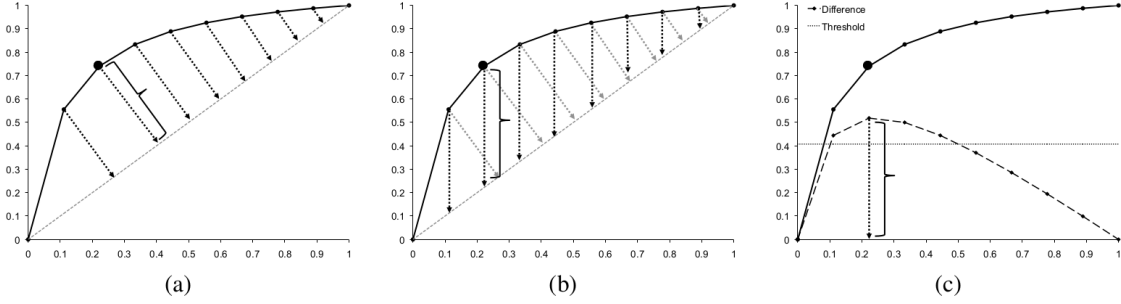


Figure 3.5: Kneedle algorithm for online knee detection. (a) depicts the smoothed and normalized data, with dashed bars indicating the perpendicular distance from $y = x$ with the maximum distance indicated. (b) shows the same data, but this time the dashed bars are rotated $\theta = 45$ degrees. The magnitude of these bars correspond to the difference values used in Kneedle. (c) shows the plot of these difference values and the corresponding threshold values. The knee is found at $x = 0.22$ and is detected $eps = 0.55$, from [83].

way to determine the $minPts$ value for DBSCAN in the literature. However, we present a stochastic general guideline for mathematically choosing the optimal $minPts$ and subsequently eps value.

3.1.4 Evaluation Metrics

In general, clustering validation can be categorized into two classes, external clustering validation, and internal clustering validation. In this thesis, we focus on internal clustering validation, which is the method we should use when there is no ground true label of data.

Let $C = \{c_1, c_2, \dots, c_q\}$ denote a set of non-overlapping and non-empty clusters where $\forall_{i \neq j} (c_i \cap c_j = \emptyset)$ and $c_i \neq \emptyset$. If the ground truth labels are not known or the supplementary variable is not available, we can evaluate how well the clustering has been performed by several metrics.

3.1.4.1 Calinski-Harabaz Index (CHI)

The Calinski-Harabaz [84] index also known as the *Variance Ratio Criterion*, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters, the higher the score, the better the performances.

The Calinski-Harabaz index $CHI(q)$ for q cluster calculated:

$$CHI(q) = \frac{tr(B_q)}{tr(W_q)} \times \frac{n - q}{q - 1} \quad (3.1)$$

Where $tr(B_q)$ is the trace between the group dispersion matrix and $tr(W_q)$ is the trace of the within cluster dispersion matrix defined by:

3. Introduction to Clustering Analysis Pipeline and Components

$$W_q = \sum_{i=1}^q \sum_{x \in c_i} (x - C_i)(x - C_i)^T \quad (3.2)$$

$$B_q = \sum_{i=1}^q n_i (C_i - C_E)(C_i - C_E)^T \quad (3.3)$$

Where n is the number of points in our dataset, c_i , C_i and n_i are the set of points in cluster, the center of cluster and the number of points in cluster i_{th} respectively. The C_E denotes the overall centroid of the dataset.

The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster and the score is fast to compute. But there is one issue with CHI. It is generally higher for convex clusters than arbitrary shaped clusters like those obtained through DBSCAN. In a convex cluster, you can draw a straight line from any point in the cluster to any other point in the cluster without leaving the cluster. For example, a U -shaped cluster would not be convex since you could not draw a straight line from one end of the U to the other without leaving the cluster and crossing across empty space. Consequently, in the case of DBSCAN, it is not significant as well as another index in the compression since DBSCAN can identify the arbitrary shaped clusters.

3.1.4.2 Davies-Bouldin Index (DBI)

The Davies-Bouldin index [85] signifies the average *similarity* between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves. A lower Davies-Bouldin index relates to a model with better separation between the clusters.

The DBI is defined as the average similarity between each cluster $c_i \in C$ and its most similar $c_j \in C$. In the context of this index, the similarity is defined as follows:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \quad (3.4)$$

Where s_i denotes the average distance between each point of cluster i_{th} and the centroid of that cluster which is also known as a cluster diameter, and the d_{ij} denotes the distance between cluster centroids i and j .

Then the Davies-Bouldin index is defines as:

$$DBI(q) = \frac{1}{q} \sum_{i=1}^q \max_{i \neq j} R_{ij} \quad (3.5)$$

Although, the index is computed only based on quantities and features inherent to the dataset, its usage of centroid distance limits the distance metric to Euclidean

3. Introduction to Clustering Analysis Pipeline and Components

space. DBI is also higher for convex clusters than arbitrary shaped clusters like those obtained through DBSCAN. Consequently, in the case of DBSCAN, it is not significant as well as another index in the compression since DBSCAN can identify the arbitrary shaped clusters.

3.1.4.3 Silhouette Index (SI)

The Silhouette index [86] is a ratio-type index that is based on silhouette values for every entity x_i measuring how well x_i fits into the cluster to which it is assigned, by comparing the within-cluster cohesion, based on the distance to all entities in the same cluster, to the cluster separation:

$$SI(x_i) = \frac{b(x_i) - a(x_i)}{\max(b(x_i), a(x_i))} \quad (3.6)$$

Where $a(x_i)$ is the distance of x_i to its own cluster, which is defined as the average distance of x_i to all the other data points in its own cluster h as:

$$a(x_i) = \frac{\sum_{j=1}^n w_{jh} d_E(x_i, x_j)}{n_h - 1}, i \neq j \quad (3.7)$$

where $d_E(i, j)$ is the squared Euclidean distance, and n_h is the number of data points in the cluster h . w_{jh} is the indicator function, which equals to 1 when x_i is in c_h and 0 when x_i is not in c_h . $b(i)$ is the distance of x_i to its closest neighbouring cluster, which is defined as the average distance of x_i to all the data points in its closest neighbouring cluster as:

$$b(x_i) = \min_{h \neq l} \left(\frac{\sum_{j=1}^n w_{jl} d_E(x_i, x_j)}{n_l} \right) \quad (3.8)$$

As a result, $SI(x_i) \in [-1, 1]$. If $SI(x_i)$ is around zero, the entity could be assigned to another cluster without making cluster cohesion or separation any worse. A negative $SI(x_i)$ suggests that x_i 's cluster assignment is damaging to cluster cohesion and separation, whereas an $SI(x_i)$ closer to 1 means the opposite. We can then quantify the validity of the whole clustering by the Silhouette index, defined as:

$$\overline{SI} = \frac{1}{n} \sum_{i \in X} SI(x_i) \quad (3.9)$$

In [8], particularly in the parallel performance analysis scenario, they made an analogy between the sequences of different actions a parallel Single Program Multiple

3. Introduction to Clustering Analysis Pipeline and Components

Data (SPMD) application performs and biological sequences such as DNA or proteins. Then, they use Multiple Sequence Alignment algorithm to quantitatively measure how the application follows the SPMD pattern which called SPMDiness. They introduce the Cluster Sequence Score as a score that evaluates the SPMDiness of a computation structure characterization obtained by using a cluster analysis. This score is also applicable to evaluate the SPMDiness of any other structural characterization. However, it is useful for in the parallel performance analysis scenario.

The literature indicates that there is no sole cluster validity index with a clear advantage over the others in every case [87]. However, the Silhouette width index has performed well in many comparative experiments [88, 89]. The Silhouette index can work with any distance measure. In the present work, we use the average Silhouette Coefficient as an evaluation metric to tune the DBSCAN hyperparameters throughout our machine learning pipeline. We apply the Silhouette index by using general Minkowski distances including the Euclidean and Manhattan distances, in line with using general Minkowski distances in our method. Furthermore, we use the Silhouette index instead of SPMDiness to keep the generality of our pipeline.

3.2 Machine Learning Pipeline

In general, the objective of Machine Learning and artificial intelligence to automate tasks and take better decisions. Building machine intelligence starts from Machine Learning concepts to implementing and building models and using them in the real world. Machine intelligence can be built using non-traditional or toiler-made computing components. A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment. In this section, we introduce an end-to-end Machine Learning pipeline based on the CRISP-DM [90] model, which will help us enhance and automatize the cluster analysis process by building machine intelligence using a structured process.

3.2.1 Machine Learning Pipelines

A Machine Learning pipeline will mainly consist of elements related to data retrieval and extraction, preparation, modeling, evaluation, and extracting insight. Figure 3.6 shows a high-level overview of a standard Machine Learning pipeline with the major phases highlighted in their blocks.

The major steps in the pipeline are briefly mentioned here:

3. Introduction to Clustering Analysis Pipeline and Components

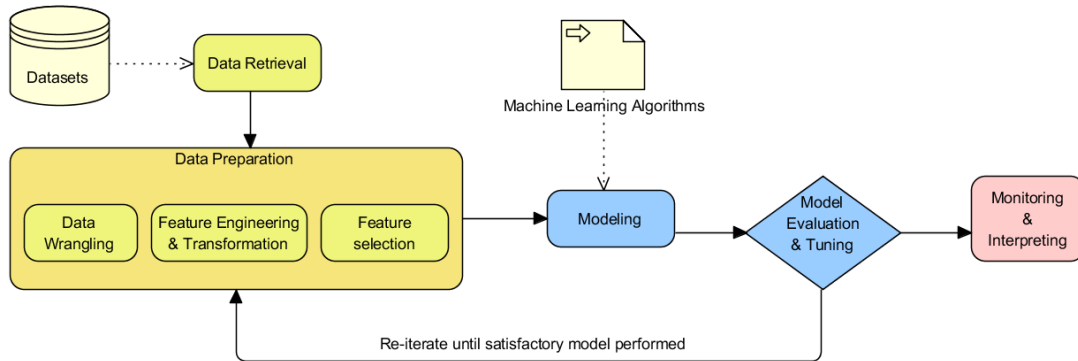


Figure 3.6: A standard Machine Learning pipeline, from [91].

- *Data retrieval:* is mainly data collection, extraction, and acquisition from various data sources and data stores.
- *Data preparation:* In this step, we preprocess the data, clean it, wrangle it, and manipulate it as needed. Initial exploratory data analysis is also carried out. The next steps involve extracting, engineering, and selecting features from the data.
 - *Data wrangling:* Mainly concerned with data processing, cleaning, munging, wrangling, and performing initial descriptive and exploratory data analysis
 - *Feature engineering and transformation:* Here, we extract important features from the raw data and even create or engineer new features from existing features which implies a certain objective preconception of solution. Furthermore, data features often need to be transformed (e.g. normalized, scaled, etc.) to prevent Machine Learning algorithms from getting biased.
 - *Feature selection:* often we need to select a subset of all available features based on feature importance and quality.
- *Modeling:* In the process of modeling, we usually feed the data features to a Machine Learning method or algorithm and train the model, typically to optimize a specific cost function of a predictive model or generalize the representations learned from the data by a descriptive model.
- *Model evaluation and tuning:* Built models are evaluated and tested on the validation datasets and, based on metrics like accuracy, F1 score, and others, the model performance is evaluated. Models have various parameters that are tuned in a process called hyperparameter optimization to get models with the best and optimal results.

3. Introduction to Clustering Analysis Pipeline and Components

- *Monitoring and interpreting:* Selected models are used to extract insights and are constantly monitored based on their predictions and results.

3.2.2 Hyperparameters Tuning

Hyperparameter tuning takes advantage of the process to test different hyperparameter configurations when training your model. It can give you optimized values for hyperparameters, which maximizes your model's predictive accuracy.

3.2.2.1 Hyperparameters

Hyperparameters contain the data that govern the training process itself. The model parameters are optimized by the training process: you run data through the operations of the model, evaluate the model goodness of fit, and adjust until you find the best values. Note that the goodness of fit of a model explains how well it matches a set of observations. Hyperparameters are tuned by running the whole training job, looking at the aggregate accuracy, and adjusting. In both cases, you are modifying the composition of your model in an effort to find the best combination to handle your problem.

Machine learning models also have parameters, which are the internal coefficients set by training or optimizing the model on a training dataset. Parameters are different from hyperparameters. Parameters are learned automatically; hyperparameters are set manually to help guide the learning process.

3.2.2.2 Hyperparameter Optimization

As such, it is often required to search for a set of hyperparameters that result in the best performance of a model on a dataset. This is called hyperparameter optimization, hyperparameter tuning, or hyperparameter search.

Without an automated technology for hyperparameter tuning, you need to make manual adjustments to the hyperparameters throughout many training runs to arrive at the optimal values. Hyperparameter tuning makes the process of determining the best hyperparameter settings easier and less tedious. An optimization procedure involves:

- *Search space:* that is a volume to be searched where each dimension represents a hyperparameter and each point represents one model configuration. Guesswork is necessary to specify the min and max values for each hyperparameter.
- *Validation Matrices:* These are tied to ML tasks. In the lecturer exist several matrices to evaluate the supervised algorithms and some evaluation matrices for unsupervised algorithms. For example, the performance of classification of the binary class is measured using Accuracy, AUROC, and Log-loss.

3. Introduction to Clustering Analysis Pipeline and Components

- *Evaluation Mechanism:* These are tied to ML tasks. In the lecturer, several matrices exist to evaluate the supervised algorithms and some matrices for unsupervised algorithms. For example, the performance of classification of the binary class can be measured by using Accuracy, AUROC, and Log-loss. Also the performance of clustering can be measured by Calinski-Harabaz Index, Davies-Bouldin Index, and Silhouette Index.
- *Hyperparameters search:* Grid search picks out a grid of hyperparameter values and evaluates all of them, and Random search randomly values a random sample of points on the search space if the search space is large [92]. It is more efficient than grid search. Smart hyperparameter tuning algorithms, such as Bayesian Optimization [93] and Evolutionary Optimization [94], pick a few hyperparameter settings, evaluate the validation matrices, adjust the hyperparameters, and re-evaluate the validation matrices.

3.2.3 Unsupervised Machine Learning Pipeline

Tuning hyperparameters for unsupervised learning problems is difficult in general since labels are not available, choosing a criterion for evaluation and in general, a method for selecting hyperparameters is not easy. However, the success of most clustering methods depends heavily on the correct choice of the involved hyperparameters.

Specifically, the clustering algorithm hyperparameter tuning is a considerable challenge when applying a clustering solution to real-world problems. Multiple iterations and considerable domain knowledge is often required to find an optimal algorithm configuration, and the process is often long and tedious [95]. In supervised problems, where ground truth is available, hyperparameter tuning is often automated, however, automated hyperparameter tuning requires accurate and objective evaluation metrics. As evaluating clustering algorithms is a considerable problem, completely automated methods of hyperparameter tuning for clustering algorithms often rely on internal evaluation metrics [96, 97], or having some ground truth labels available for external evaluation metrics [98, 99], which moves the problem into the semi-automated space.

However, these methods of evaluation are often flawed, and cannot comment on the quality of the clusters developed for the use case. Internal methods measure the cluster quality with similarity metrics and tend to be biased towards particular types of clustering algorithms [100]. Another method of evaluation is through meta-criteria, such as stability and statistical significance, which can be useful in determining the quality of a clustering algorithm but less so in comparing the results of multiple algorithms. In [100], they asserted that clustering algorithms cannot be evaluated

independently of the context in which they will be used. Domain-specific evaluation can be highly subjective and often requires significant time and resources to perform. As the effect of hyperparameters on clustering results cannot be described through a convex function, an exhaustive grid search is required to find the optimal hyperparameters [96]. For an individual to manually perform an exhaustive grid search and evaluate all of the possible results would be a time-intensive and cumbersome process.

We introduce a framework for automated hyperparameter tuning of clustering problems, using mean silhouette score as an internal metric [13] and exhaustive grid search [101]. The contribution of this dissertation is that a framework for the automated hyperparameter tuning of a clustering problem is presented and evaluated on a real-world clustering problem in the specific domain of performance analysis for HPC programs.

3.3 Feature Selection

The advent of high-dimensional data has brought unprecedented challenges to machine learning researchers, making the learning task more complex and computationally demanding. The term high dimensionality is applied to a database that presents one of the following characteristics: (a) the number of samples is very high; (b) the number of features is very high, or (c) both the number of samples and features are very high.

When dealing with high-dimensional data, learning algorithms can degenerate their performance due to over-fitting, learned models decrease their interpretability as they are more complex, and finally speed and efficiency of the algorithms decline in accordance with size. Machine learning can take advantage of feature selection methods to be able to reduce the dimensionality of a given problem. Feature selection is the process of detecting the relevant features and discarding the irrelevant and redundant ones, intending to obtain a small subset of features that describes properly the given problem with a minimum degradation or even improvement in performance [102].

Since feature selection maintains the original features, it is especially useful for applications where the original features are important for model interpreting and knowledge extraction [103].

In the case of unsupervised learning, there are some issues and limitations that are made important "Irrelevant feature removing": degrade learning quality; consume more memory, computational time, and the visualization lifetimes. With an appropriate feature subset selection, we can archive those limitations.

The feature selection step becomes more important when unsupervised learning (Clustering) is used in high-dimensional data. The cluster structure of interest to domain experts can often be best described using a subset of attributes. The

3. Introduction to Clustering Analysis Pipeline and Components

inclusion of other attributes can degrade clustering performance and complicate cluster interpretation. Compared to methods based on all the variables, a superior feature selection method consistently yields more accurate estimates of the number of clusters, such as more parsimonious clustering models and easier visualization of results.

In this dissertation, we are interested in the feature selection approach for cluster analysis. Most clustering methods assume all features to be equally important for clustering, or in other words, they do not distinguish among different features. This is one of the reasons why most clustering algorithms may not perform well in the face of high-dimensional data. The appropriate feature helps in creating clusters while an inappropriate feature may not help in creating clusters and, on contrary; it might be efficient the clustering algorithms adversely by blurring the clusters. Unimportant features are noisy or irrelevant and can be removed to reduce the data size for more efficient clustering. To continue, we address the methods of selecting a subset of important features.

This chapter will present the foundations of feature selection, as well as a description of some existing feature selection methods.

3.3.1 Foundations of Feature Selection

Feature selection can be defined as the process of detecting the relevant features and discarding the irrelevant and redundant ones to obtain a subset of features that describes properly the given problem with a minimum degradation of performance. It has several advantages [103]:

- Improving the performance of the machine learning algorithms.
- Data understanding, gaining knowledge about the process and perhaps helping to visualize it.
- General data reduction, limiting storage requirements and perhaps helping in reducing costs.
- Feature set reduction, saving resources in the next round of data collection or during utilization.
- Simplicity, the possibility of using simpler models and gaining speed.

3.3.1.1 Feature relevance

Intuitively, it can be determined that a feature is relevant if it contains some information about the latent data structure. More formally, we can classify features into three disjoint categories, namely, strongly relevant, weakly relevant, and irrelevant features [104].

The strong relevance of a feature indicates that the feature is always necessary; it cannot be removed without affecting the machine learning model quality. Weak relevance suggests that the feature is not always necessary but may become necessary for an optimal subset at certain conditions. Irrelevance indicates that the feature is not necessary at all. An optimal subset should include all strongly relevant features, none of the irrelevant features, and a subset of weakly relevant features. However, it is not given in the definitions which weakly relevant features should be selected and which of them removed. Therefore, it is necessary to define feature redundancy among relevant features.

3.3.1.2 Feature redundancy

A feature is usually considered redundant in terms of feature correlation [105]. It is widely accepted that two features are redundant to each other if their values are completely correlated, but it might not be so easy to determine feature redundancy when a feature is correlated with a set of features. Consequently, a redundant feature should be removed if it is weakly relevant when the feature has a Markov blanket within the current set of features [106]. Since irrelevant features should be removed anyway, they are excluded from this definition of redundant features.

3.3.2 Feature selection methods

Feature selection methods can be divided according to two approaches: individual evaluation and subset evaluation [106]. Individual evaluation is also known as feature ranking and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one concerning this measure. While the individual evaluation is incapable of removing redundant features because redundant features are likely to have similar rankings, the subset evaluation approach can group the redundant features and then select the most relevant ones. However, methods in this framework can suffer from an inevitable problem caused by searching through all feature subsets required in the subset generation step, and thus, both approaches are worth to be studied.

3. Introduction to Clustering Analysis Pipeline and Components

Aside from this classification, three major approaches can be distinguished based upon the relationship between a feature selection algorithm and the inductive learning method used to infer a model [103]:

- *Filters*, which rely on the general characteristics of training data and carry out the feature selection process as a preprocessing step with independence of the induction algorithm. This model is advantageous for its low computational cost and good generalization ability.
- *Wrappers*, which involve a learning algorithm as a black box and consists of using its prediction performance to assess the relative usefulness of subsets of variables. In other words, the feature selection algorithm uses the learning method as a subroutine with the computational burden that comes from calling the learning algorithm to evaluate each subset of features. However, this iteration with the classifier tends to give better performance results than filters.
- *Embedded*, which perform feature selection in the process of training and are usually specific to given learning machines. Therefore, the search for an optimal subset of features is built into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. This approach can capture dependencies at a lower computational cost than wrappers.

3.3.2.1 Filter methods

Filter methods are based on performance evaluation metrics calculated directly from the data, without direct feedback from predictors that will finally be used on data with the reduced number of features [103]. As mentioned above, these algorithms are usually computationally less expensive than wrappers or embedded methods. In general, it consists of algorithms that are built in the adaptive systems for data analysis, see Figure 3.7.

Thabtah et al. [107] introduced an observed frequency-based feature selection method called Least Lost (L^2) to reduce the dimensionality of data by eliminating noisy data from the datasets while maintaining a healthy classifier performance. It is a simplified and in-built approach that involves the ranking of each variable in ascending order based on the L^2 distance between observed and expected variables and class labels. The scores are computed based on the observed and expected probabilities of the available features. Tests conducted using datasets from the University of Irvine Repository (UCI) reported that L^2 , when applied in the preprocessing phase, results in fewer features being obtained. When these are further processed by a machine learning algorithm, they derive competitive classifiers in terms of accuracy.

3. Introduction to Clustering Analysis Pipeline and Components

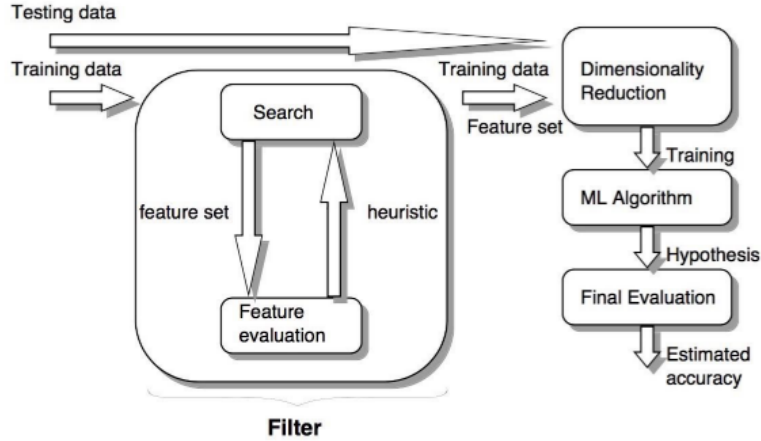


Figure 3.7: Generic Filter base Feature Selection Algorithm.

In [108], they proposed the redundant penalty between the feature mutual information algorithm (RPFMI). It is a filter-based feature selection algorithm to identify optimal features in terms of redundancy. It considers three factors to select the features: the redundancy between features, the impact between selected features and classes and the relationship between candidate features and classes. However, it still needs the labels to select the features.

Gao et al. [109] introduced the dynamic change of selected feature (DCSF) which takes dynamic information changes of the selected features with the class labels into account in the feature selection process; this to yield more accurate and efficient results. This novel model uses conditional mutual information between candidate features and class labels to identify the most informative features; the other conventional filter methods use mutual information to compute the relevancy of the candidate features to the select optimal feature subset. The experimental results implied that DCSF has the highest average classification accuracy of all the other compared methods.

Another filter mechanism presented in [110] which is quite unique. These authors focus on selecting features based on their true rankings obtained by applying ReliefF [111] and Fisher Score [112] rather than focusing on their mutual redundancies. MIRFFS (Mutual Information, ReliefF, and Fisher Score), the proposed mechanism, used differential evolution (DE) [113], as the search strategy and it has two parts: one mechanism to be applied on single-objective problems and the other on multi-objective problems.

In [114], they introduced multivariate relative discrimination criterion (MRDC), a filter-based feature selection mechanism to enhance the performance of the text classification process. This is accomplished by diminishing the dimensionality in feature space using minimal-redundancy and maximal-relevancy (mRmR) [115]. MRDC involves identifying the most relevant features using relative discrimination criterion

3. Introduction to Clustering Analysis Pipeline and Components

(RDC) [116]. Since RDC is not capable of classifying the irrelevant features, it utilizes the Pearson correlation matrix to perform that task.

[117] used three robust filter methods in combination to produce a feature selection mechanism (vectors of scores/ V -score) to select the most relevant features of a given dataset while eliminating the shortcomings and maximizing the advantages. They used information gain [118], chi-squared statistic [119], and inter-correlation methods (CFS) [120] together to stabilize each feature's ranking score; they were able to reap more accurate prediction results rather than when applying them individually.

OSFSMI (Online Stream Feature Selection Method based on Mutual Information) and OSFSMI- k is another mutual information-based online streaming feature selection method, presented by [121], to distinguish between the most informative and uninformative features. This is done by computing the correlation between features and their relevancy, to the class labels where the number of instances increases exponentially (for example, social networks, finance analysis applications, and traffic network monitoring systems). The general framework followed by the proposed OSFSMI model comprises two unique phases: online relevancy analysis to compute the relevancy of each newly arriving feature, and online redundancy analysis to estimate the effectiveness of each selected feature and eliminate any with effectiveness below the average. OSFSMI- k is a modified version of OSFSMI, developed to address the issues arising due to the continuously increasing nature of features. To end this, OSFSMI- k keeps selecting the correlated features until the size of the selected feature subset reaches a constant value (k).

In [122], they proposed a normalized mutual information feature selection (NMIFS), to evaluate the relevancy and redundancy in the features of a given dataset. Researchers have used three mutual information-based feature selection methods: Battiti's mutual information feature selector (MIFS), MIFS-U (Battiti, 1994), and min-redundancy max-relevance (mRMR) [115] criteria to develop NMIFS by enhancing their strengths and minimizing their weaknesses. They also present the Genetic algorithm, guided by mutual information for feature selection (GAMIFS), a hybrid version of both the filter and wrapper methods that combines NMIFS and genetic algorithms to fine-tune their performance.

3.3.2.2 Wrapper methods

The existing algorithms belonging to the wrapper model utilize a machine learning algorithm (Generally classification) as a selection criterion. The algorithms of this method are wrapped around the adaptive systems providing them subsets of features and receiving their feedback (usually accuracy). These wrapper approaches are aimed at improving, the results of the specific predictors they work with. The wrapper model

3. Introduction to Clustering Analysis Pipeline and Components

utilizes a machine-learning algorithm to evaluate the quality of selected features. Figure 3.8 shows the generic wrapper algorithm which starts to find a subset of features. Then, it evaluates the machine-learning algorithm quality using the selected subset. Finally, it repeats two previously aforementioned steps until they converge to the desired quality.

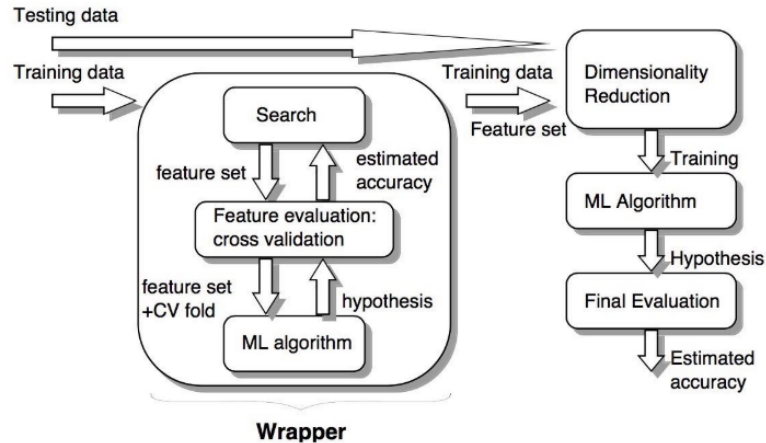


Figure 3.8: Generic Wrapper feature selectors algorithm.

Evaluating all possible subsets of features is impossible in high-dimensional datasets. Therefore, a heuristic search strategy is adopted to reduce the search space. The wrapper model is very computationally expensive compared to the filter model. Yet, it produces better clustering since we aim to select features that maximize the quality of the of the specific machine learning algorithm. It is still biased toward the used clustering method. Different wrapper feature selection methods for clustering were proposed by changing the combination of the search strategy and the utilized clustering algorithm. The method proposed in [115] is an example of a wrapper that involves maximum likelihood criteria and feature selection and the mixture of Gaussians as a clustering method. Others use conventional clustering methods such as k-means and any search strategy as feature selector [123].

Weka [124] provides the Wrapper SubsetEval method, which evaluates attribute sets by using a learning scheme. Cross-validation is used to estimate the accuracy of the learning scheme for a set of attributes. The algorithm starts with the empty set of attributes and searches forward, adding attributes until performance does not improve further.

However, they exhibit some limitations. The first, and probably one of the most important deficits is the lack of a more extensive empirical evaluation of the methods and, in particular, a comparison between filters and wrappers. A second shortcoming is that many of these approaches are focused on classification, and there is no theoretical or experimental evidence related to their behavior in clustering analysis. In this

3. Introduction to Clustering Analysis Pipeline and Components

thesis, we use the leveraging of the Independent Component Analysis to propose a filter-based feature selection method for the clustering.

The wrapper method works usually associated with learning algorithms, or specifically, predictive models in most cases. In [125, 126], the selected features firstly train a predictive model and then are evaluated on a fixed validation set. Other criteria like least square, spectral theory, and sparse regularize, have been recently proposed for this problem [127]. Any feature selection method achieves a proportion of information from the data. In this thesis, we have been interested in feature selection approach for the unlabeled and noisy datasets, and particularly for the performance data.

3.3.2.3 Other approaches

There exist numerous papers and books proving the benefits of the feature selection process [128, 129]. However, most researchers agree that there is not a so-called “best method” and their efforts are focused on finding a good method for a specific problem set. Therefore, the recent feature selection methods are constantly appearing using different strategies: a) combining several feature selection methods, which could be done by using algorithms from the same approach, such as two filters [130], or coordinating algorithms from two different approaches, usually, filters and wrappers [131, 132] b) combining feature selection approaches with other techniques, such as feature extraction [133] or tree ensembles [134]; c) reinterpreting existing algorithms [135], sometimes to adapt them to specific problems [136]; d) creating methods to deal with still unresolved situations [137, 138] using an ensemble of feature selection techniques to ensure a better behavior [139, 140]).

To confront the problem of the high dimensionality of data, feature selection algorithms have become indispensable components of the learning process. Hence, a correct selection of the features can lead to an improvement of the inductive learner, either in terms of learning speed, generalization capacity, or simplicity of the induced model.

3.4 Feature Transformation

Features that are on a continuous scale are subject to introduce the potential issues that we may have to confront. Some of the problems that are prevalent with continuous features can be mitigated through the type of model that we choose. For example, models that construct relationships between the features and the response variable that are based on the rank of the features values rather than the actual value, like trees, are immune to features distributions that are skewed or to individual samples that have unusual values (i.e., outliers). Other models such as K-nearest neighbors and

3. Introduction to Clustering Analysis Pipeline and Components

support vector machines are much more sensitive to features with skewed distributions or outliers. Continuous features that are highly correlated with each other are another regularly occurring scenario that presents a problem for some models but not for others. Partial least squares, for instance, are specifically built to directly handle highly correlated features. But models like multiple linear regression or neural networks are adversely affected in this situation [141].

If we desire to utilize and explore the goodness-of-fit of more types of machine learning models, the issues presented by the features need to be addressed by transforming them in a useful way.

On the same hand, the majority of density-based clustering algorithms such as DBSCAN fail to properly find clusters in data exposing different densities in various regions of the feature space. This failure results from using a single global density threshold on all the data points.

In this thesis, we propose a parametric multilinear transformation method to homogenize cluster densities while preserving the topological structure of the dataset. We use this transformation to improve the goodness-of-fit of the density-based algorithms.

In this chapter, we will provide a brief background of transformation approaches and illustrate how to handle continuous features with commonly occurring issues including vastly different scales, follow a skewed distribution, contain a small number of extreme values, etc. The techniques discussed thus far are unsupervised, meaning that the objective of the dimension reduction is based solely on the features.

3.4.1 Individual Feature Transformation

There are a variety of modifications that can be made to an individual feature that might improve its utility in a model. The first type of transformations to a single feature discussed here are those that change the scale of the data.

The normal distribution has a fundamental role in the statistical literature since it forms the basis of most of the statistical methods such as regression analysis, analysis of variance, and *t-test*. Therefore, the validity of the related results necessitates the agreement between the distribution of the observed data and this theoretical distribution. In cases where this agreement deteriorates, which is common in real-life datasets, transformation methods might be a practical remedy to secure it. The most popular and commonly used method is the *Box-Cox* power transformation [142]. Since its proposition, it has been applied in various fields. Some of the recent works include Lee et al. [143], Gillard [144] and Sun et al. [145]. *Box-Cox* transformation mainly applies a deterministic power function to the raw data by using the estimate of the power transformation parameter, λ . Therefore, the estimation of λ is crucial. The

3. Introduction to Clustering Analysis Pipeline and Components

original proposal of the methodology [142] involved the maximum likelihood estimation (MLE). Alternative methodologies included the works of Rahman and Pearson [146], Osborne [147], and Dag et al. [148]. Whereas the first two studies proposed the estimation of λ via two normality tests, specifically Shapiro-Wilk and Anderson-Darling tests, respectively, the third one proposed simulating a single artificial and non-informative co-variate and finding which linear regression model minimizes the sum of squared error among several simple linear regression models. These studies showed that the MLE of λ might be biased and inefficient.

Another important transformation to an individual feature is for features that have values bounded between zero and one, such as proportions. The problem with modeling this type of outcome is that model predictions might not be guaranteed to be within the same boundaries. For data between zero and one, the *logit* transformation [149] could be used. If x is the feature, the *logit* transformation is $\text{logit}(p) = \log(x/1 - p)$.

This transformation changes the scale from values between zero and one to values between negative and positive infinity. On the extremes, when the data are absolute zero or one, a small constant can be added or subtracted to avoid division by zero. Once model predictions are created, the inverse logit transformation can be used to place the values back on their original scale. An alternative to the logit transformation is the *arcsine* transformation [150]. This is primarily used on the square root of the proportions (e.g., $\hat{p} = \text{arcsine}(\sqrt{p})$).

Another common technique for modifying the scale of a feature is to standardize its value in order to have specific properties. *Centering* a feature is a common technique. The feature's training set average is subtracted from the feature's values. When this is applied separately to each variable, the collection of variables would have a common mean value (i.e., zero). Similarly, *scaling* is the process of dividing a variable by the corresponding training set's standard deviation. This ensures that variables have a standard deviation of one. Alternatively, *range scaling* uses the training set minimum and maximum values to translate the data to be within an arbitrary range (usually zero and one). Again, it is emphasized that the statistics required for the transformation (e.g., the mean) are estimated from the training set and are applied to all data sets (e.g., the test set or new samples) [151].

These transformations are mostly innocuous and are typically needed when the model requires the features to be in common units. For example, when the distance or dot products between features are used (such as K-nearest neighbors or support vector machines) or when the variables are required to be a common scale in order to apply a penalty (e.g., the lasso or ridge regression), a standardization procedure is essential.

Another helpful transformation technique that can be used on data containing a time or sequence effect is simple data smoothing. For example, a rolling average

can be used to reduce excess noise in the feature or outcome data before modeling. For example, a running 5-point mean would replace each data point with the average of itself and the two data points before and after its position. As one might expect, the size of the moving window is important; too large and the smoothing effect can eliminate important trends, such as nonlinear patterns.

A short-running median can also be helpful, especially if there are significant outliers. When an outlier falls into a moving window, the mean value is pulled towards the outlier. The median would be very insensitive to an aberrant value and is a better choice. It also has the effect of changing fewer of the original data points. In addition, other smoothers techniques can be used, such as smoothing splines. However, the simplicity and robustness of a short-running median can be an attractive approach to this type of data.

3.4.2 Multi-Feature Transformation

When creating new features from multiple features, there is a possibility of correcting a variety of issues such as outliers or collinearity. It can also help reduce the dimensionality of the feature space in ways that might improve performance and reduce computational time for models.

Linear projection methods have been shown to effectively identify meaningful projections of the original features. These methods are linear in the sense that they take a matrix X of numeric features values and create new components that are linear combinations of the original data. If there are n data points and m features, the components could be denoted as the $n \times m$ matrix \hat{X} and are created using $\hat{X} = XA$ where the $m \times m$ matrix A is often called the projection matrix. The difference between the projection methods is in how the projection values are determined.

Principal component analysis (PCA) [152] is still one of the most widely used linear projection methods. Specifically, the objective of PCA is to find linear combinations of the original features such that the combinations summarize the maximal amount of variation in the original feature space. Simultaneously, the new PCA components are required to be orthogonal (i.e., uncorrelated) to each other. The property of orthogonality enables the feature space variability to be neatly partitioned in a way that does not overlap.

PCA is a particularly useful tool when the available data are composed of one or more clusters of features that contain redundant information (e.g., features that are highly correlated with one another). An important side benefit of this technique is that the resulting PCA scores are uncorrelated. This property is very useful for modeling techniques (e.g., multiple linear regression, neural networks, support

3. Introduction to Clustering Analysis Pipeline and Components

vector machines, and others) that need the features to be relatively uncorrelated. Although, Principal component analysis is an effective technique when features are linearly correlated and when the resulting scores are associated with the response, the orthogonal partitioning of the feature space may not provide a good predictive relationship with the response, especially if the true underlying relationship between the features and the response is non-linear.

kernel PCA [153] method uses a nonlinear basis expansion. The *kernel PCA* approach combines a specific mathematical view of PCA with kernel functions and the *kernel* ‘trick’ to enable PCA to expand the dimension of the feature space in which dimension reduction is performed (not unlike basis expansions). For example, we can directly expand a linear combination as a polynomial of degree d by using the polynomial kernel. The RBF kernel is also known as the Gaussian kernel due to its mathematical form and its relationship to the normal distribution. The radial basis and polynomial kernels are popular starting points for kernel PCA.

Independent component analysis (ICA) [154] creates new components that are linear combinations of the original features but does so in a way that the components are as statistically independent from one another as possible. This enables ICA to be able to model a broader set of trends than PCA, which focuses on orthogonality and linear relationships. ICA methods need to meet the constraint of statistical independence to maximize the “non-Gaussianity” of the resulting components. For example, the fastICA [155] approach uses an information theory called *negentropy* to measure non-Gaussianity. In practice, ICA should create components that are dissimilar from PCA unless the features demonstrate significant multivariate normality or strictly linear trends. Also, unlike PCA, there is no unique ordering of the components. It is common to normalize and whiten the data before running the ICA calculations. Whitening, in this case, means converting the original values to the full set of PCA components. Also, the ICA computations are typically initialized with random parameter values and the result can be sensitive to these values.

Non-negative matrix factorization [156] is another linear projection method that is specific to features that are greater than or equal to zero. In this case, the algorithm finds the coefficients of A such that their values are also non-negative (thus ensuring that the new features have the same property). This approach is popular for text data where features are word counts, imaging, and biological measures (e.g., the amount of RNA expressed for a specific gene).

The method for determining the coefficients is conceptually simple: find the best set of coefficients that make the scores as “close” as possible to the original data with the constraint of non-negativity. Closeness can be measured using mean squared error (aggregated across the features) or other measures such as the *Kullback–Leibler*

3. Introduction to Clustering Analysis Pipeline and Components

divergence [157]. The latter is an information-theoretic method to measure the distance between two probability distributions. Like ICA, the numerical solver is initialized using random numbers and the final solution can be sensitive to these values and the order of the components is arbitrary

Autoencoders are computationally complex multivariate methods for finding representations of the feature data and are commonly used in deep learning models [158]. The idea is to create a nonlinear mapping between the original feature data and a set of artificial features (that is usually the same size). These new features, which may not have any sensible interpretation, are then used as the model features. While this does sound very similar to the previous projection methods, autoencoders are very different in terms of how the new features are derived and also in their potential benefit.

The *spatial sign* transformation takes a set of features and transforms them in a way that the new values have the same distance to the center of the distribution [159]. This approach is also referred to as *global contrast normalization* [160] and is often used with image analysis to prevent images from having different levels of contrast in different parts of images.

3.4.2.1 Summery

Numeric features, in their raw form, may or may not be in an effective form that allows a model to find a good relationship with the response. For example, a feature may be measured, but its squared version is what is truly related to the response. Straightforward transformations such as centering, scaling, or transforming a distribution to symmetry are necessary steps for some models to be able to identify the predictive signal. Other transformations such as basis expansions and splines can translate a feature in its original scale to nonlinear scales that may be informative.

Instead of expanding the feature space, it may be necessary to enrich the existing feaster space. This can be accomplished using unsupervised techniques such as PCA, ICA, or NNMF.

The way to explore nonlinear relationships between features and the response is through the combination of a kernel function and PCA. This approach is very computationally efficient and enables the exploration of much larger dimensional space.

Finally, autoencoders or the spatial sign transformation offer novel engineering approaches that can harness information in unlabeled data or dampen the effect of extreme samples.

3.5 Clustering Result Interpretation

Due to the ever-growing complexity of ML models and their use in increasingly sensitive applications, it has become crucial to endow these models with the capability to explain their predictions in a way that is interpretable for a human. Explainable AI (XAI) has emerged as an important direction for machine learning, and excellent results have been reported in selected tasks such as explaining the predictions of popular DNN classifiers [161–165].

The emerging field of Explainable AI (XAI) has so far mainly focused on supervised learning, in particular, deep neural network classifiers. In many practical problems, however, the label information is not given and the goal is instead to discover the underlying structure of the data, for example, its clusters. While powerful methods exist for extracting the cluster structure in data, they typically do not answer the question of why a certain data point has been assigned to a given cluster.

Some works explore ways of merging the clustering models and neural networks to produce better and more flexible ML models. For example, deep clustering approaches typically build a clustering objective on top of deep representations [166–170]. Other models, in particular, the k-meansNet [171] design the neural network in a way that simulates a clustering model so that the learned neural networks solution can be interpreted as a clustering solution. Note that in all these works, the purpose is more to enhance a basic clustering model by providing the flexibility of neural network representation and training, whereas our work focuses on making existing popular clustering algorithms explainable.

Another set of related works focuses on the problem of learning a good clustering model, by identifying a subset of relevant features that support the cluster structure. Some methods identify relevant features by running the same clustering algorithm multiple times on different feature subsets [172]. Other approaches simultaneously solve feature selection and clustering by defining a joint objective function to be minimized [172]. While feature selection can identify the set of features required to represent the overall cluster structure, our work builds up by identifying among those features which ones are truly responsible for a given cluster or a given cluster assignment.

Further related works focus on quantitatively validating clustering solutions. Examples of validation metrics are compactness / separation of clusters [173], cluster stability under resampling / perturbations [174, 175], or purity, i.e. the absence of examples with different labels in the same cluster [176]. Our work enhances the validation of clustering models by producing human-interpretable feedback, a critical step to identify whether cluster assignments are supported by meaningful features or by what the user would consider being artifacts.

3. Introduction to Clustering Analysis Pipeline and Components

Lastly, user interfaces have been developed to better navigate cluster structures, as they occur, e.g. in biology applications [177, 178]. Also, the use of prototypes has been proposed to visualize deep image clustering models [168] or explain kernel methods for property prediction of chemical compounds [179]. Although these works produce useful and informative visualizations which may help to guide the process of clustering, extracting information from visualization is intuitive and manual.

In this thesis, we bring a novel developed explanation capabilities to clustering, a highly needed functionality, considering that in the first place one of the main motivations for performing a clustering is knowledge discovery. Especially in high-dimensional feature space, a clustering for knowledge discovery can only provide a few prototypical data points for each cluster. Such prototypes, however, do not reveal which features made them prototypical. Instead, we would like to let the clustering model explain itself in terms of the very features that have contributed to the cluster assignments. To the best of our knowledge, our work is the first-ever attempt to systematically and comprehensively obtain such explanations by exploring the cluster shape.

Part II

New Techniques to Enhance the Clustering Analysis

4

New Unsupervised Feature Selection Technique for Noisy Data

Contents

4.1	Introduction	63
4.2	Related Works	65
4.3	Background	66
4.3.1	Independent Component Analysis (ICA)	66
4.3.2	Oblique Rotation.	67
4.4	RIFS Algorithm Description	68
4.4.1	Computational Complexity Analysis	69
4.5	Empirical Study	70
4.5.1	Parameter Selection	70
4.5.2	Data Sets	70
4.5.3	Study of Unsupervised Cases	71
4.5.4	Study of Supervised Cases	75

In this chapter, we present our new Robust independent Feature Selection algorithm that can identify the most relevant feature subsets in noisy and unlabeled data, that we use to automatically reduce the dimensionality of the potentially high-dimensional data.

4.1 Introduction

Data is often represented by high dimensional feature vectors in many areas, such as face recognition, image possessing and text mining. In practice, not all features are relevant and important to the learning task, many of them are often correlated,

4. New Unsupervised Feature Selection Technique for Noisy Data



Figure 4.1: Gaussian noisy versions of the image sample from COIL20 data set with different σ^2 . From left to right σ^2 is: 0, 0.1, 0.4 and 0.7.

redundant, or even noisy sometimes, which may result in adverse effects such as over-fitting, low efficiency and poor performance. Moreover, high dimensionality significantly increases the time and space requirements for processing the data. Feature selection is one effective means to identify relevant features for dimension reduction [180]. Once a reduced feature subset is chosen, conventional data analysis techniques can then be applied.

From the perspective of label availability, feature selection algorithms can also be classified into supervised feature selection and unsupervised feature selection. Supervised feature selection methods, such as Pearson correlation coefficients [181], Fisher score [182], and Information gain [183], are usually able to effectively select good features since labels of training data, which contain the essential discriminative information for classification that can be used. However, in practice, there is usually no shortage of unlabeled data but labels are expensive. Hence, it is a great significance to develop unsupervised feature selection algorithms which can make use of all the data points. In this chapter, we consider the problem of selecting features in unsupervised learning scenario which is more challenging task because of the lack of label information that would guide the search for relevant features.

Another important factor which affects the performance of feature selection is the consideration of outliers and noise. Real data is not usually ideally distributed and outliers or noise often appear in the data, thus the traditional feature selection approach may work well on clean data. However, it is very likely to fail in noisy data sets. [184]. As an example, various types of noise are arisen during the image transmission and acquisition that Gaussian noise is one of them. It means noisy image pixel is the sum of the actual pixel value and a random Gaussian distributed noise value [185]. Fig.(4.1) shows noisy versions of sample image from COIL20 ¹ data set with different σ^2 values. It can be seen that indeed, as the σ^2 value increases, the picture gets more and more ambiguous. In experimental part of this work we aim at applying our robust feature selection on cropped data set by Gaussian noise with $\sigma^2 \leq 0.7$.

¹<http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html>

4. *New Unsupervised Feature Selection Technique for Noisy Data*

In this chapter, we introduce a new unsupervised feature selection algorithm, called Robust Independent Feature Selection (RIFS). We perform noise separation, isolation and robust feature selection simultaneously to select the most important and discriminative features for both unsupervised and supervised learning. Specifically, our purposed method exploits the structure of the latent independent components of a feature set and separates the noise as a component. By using independent component analysis, RIFS suggests a principled way to measure the similarity between different features and to rank each of them without label information. Thus, it imposes an oblige rotation on the independent factor indicator matrix to isolate the noise.

The rest of the chapter is as follow: in Section 4.2, we present a brief review of the related work. In section 4.3, the background techniques are briefly reviewed. Our proposed method, which we name Robust Independent Feature Selection (RIFS), is described in Section 4.4. The experimental results are illustrated in Section 4.5.

4.2 Related Works

Feature selection algorithms can be grouped into two main families: filter and wrapper. Filter methods [127, 186] select a subset of features by evaluating statistical properties of data. For wrapper methods [187], feature selection is wrapped in a learning algorithm and the performance on selected features is taken as the evaluation criterion. Wrapper methods couples feature selection with built-in mining algorithm tightly, which lead to less generality and extensive computation. In this work, we are particularly interested in the filter methods which are much more affordable.

The majority of the existing filter methods are supervised. Perhaps, Max variance [188] is the simplest yet effective unsupervised assessing criterion for selecting features. This measure principally projects the data points along the dimensions of maximum variances. Although the maximum variance metrics detect features that are purposeful for descriptive analysis, there is no reason to assume that these features must be useful for discriminating between data in distinct classes.

The Principal Component Analysis (PCA) algorithm shares the same principle of maximizing variance. Thus, some feature selection algorithms [189, 190] are available for selecting the features by means of Principal Component Analysis. However, its orthogonal constraint on the feature selection projection matrix is unreasonable since feature vectors are not necessarily orthogonal with each other in nature.

Currently, the Laplacian Score algorithm [186] and its extensions [127, 191] have been proposed to select features by leverage of manifold learning. Laplacian Score algorithm utilizes a spectral graph to extract the local geometric structure of the data then it selects feature subset which is mapped perfectly to the graph.

4. New Unsupervised Feature Selection Technique for Noisy Data

Another important factor which affects the performance of feature selection is the consideration of outliers and noise. In reality, outliers and noise are corrupting the distribution of the data, thus it is important or even necessary to consider noise robustness for unsupervised feature selection. Zhai,[184] purposed RUFFS method which jointly performs robust label learning via local learning regularized robust orthogonal non-negative matrix factorization and robust feature learning via joint $l_{1,2}$ -norms minimization. A remarkable drawback of the algorithm is that its performance is relatively sensitive to the number of selected features.

The intention of our work is to propose an unsupervised feature selection technique that can choose better features subset across a noisy data set; thereby, we are proposing a hybrid algorithm to utilize feature selection along with the noises separation and isolation.

4.3 Background

We consider the canonical problem of unsupervised feature selection is the following. We use X to indicate a data set of N data points $X = (x_1, x_2, \dots, x_N)$, $x_i \in R^M$. The objective is to find a feature subset with size d which includes the majority informative features. In preference to, the points $[x'_1, x'_2, \dots, x'_N]$ mirrored in the reduced d -dimensional space $x'_i \in R^d$ can perfectly maintain the original geometric structure of data in M -dimensional space.

In the remaining part of this section, we discuss the main data mining techniques that we utilize in our feature selection approach.

4.3.1 Independent Component Analysis (ICA)

To detect the latent structure of data, Independent Component Analysis (ICA) [192] tries to unmix some different sources (includes noise) that have been collected together. ICA is a statistical and computational technique for revealing the hidden sources/components that underlie sets of random variables, measurements or signals. The main ICA problem assumes that the observation X is an unknown linear mixture A of the M' unknown sources S :

$$X = AS, \quad X \in \mathfrak{R}^M, \quad A \in \mathfrak{R}^{M'}, \quad S \in \mathfrak{R}^{M' \times M}$$

We assume that each component s_i of S is zero-mean, mutually independent $p(s_i, s_j) = p(s_i)p(s_j)$ and drawn from different probability distribution which is not Gaussian except for at most one. The goal of ICA is to find an approximation W (demixing matrix) of A^{-1} such that:

$$\hat{S} = WX \approx S$$

4. New Unsupervised Feature Selection Technique for Noisy Data

ICA is a generative model since the model describes how X could be generated from A and S . ICA tries to find A by estimating the matrices of its SVD decomposition $A = U\Sigma V^T$ [193]. Ideally, W should be:

$$W = A^{-1} = V\Sigma^{-1}U^T$$

FastICA [194] is an algorithm that searches the optimal value of W , which estimates the sources S by approximating statistical independence. The algorithm starts from an initial condition, for example, random demixing weights w_0 . Then, on each iteration step, the weights w_0 are first updated by:

$$w_0^+ = E\{x(w_0^T x)^3\} - 3\|w_0\|^2 w_0$$

so that the corresponding sources become more independent, and then $w_0^+/norm$ (normalized), so that w_0 stays orthonormal. The iteration is continued until the weights converge $|w_0^T w_0^+| \approx 1$. The w_0 is an optimal approximation of W .

$$\hat{S} = w_0 X \approx S \quad (4.1)$$

When one tries to perform feature analysis of the data, each row of A can reflect the data distribution on the corresponding hidden source. Thus, if the data is cropped by noise, the noise is remarked as an independent source.

4.3.2 Oblique Rotation.

Preliminary result from a factor analysis is not easy to post-process (i.e. clustering, classification). Simply, rotation has been developed not long after factor analysis to help us to clarify and simplify the results of a factor analysis. Two main types of rotation are used: orthogonal when the new axes are also orthogonal to each other, and oblique when the new axes are not required to be orthogonal to each other. The Promax [195] is an oblique rotation technique which has the advantage of being fast and conceptually simple. Promax rotation has three distinct steps.

First, it extracts the Varimax [196] orthogonal rotated matrix $\Lambda_R = \{\lambda_{ij}\}$.

Second, a target matrix is contrived to power matrix $P = (p_{ij})_{p \times m}$ by raising the factor structure coefficients to the power of Promax rotation $k > 1$,

$$p_{ij} = \left| \frac{\lambda_{ij}}{\sqrt{\sum_{j=1}^m \lambda_{ij}^2}} \right|^{k+1} \left(\frac{\sqrt{\sum_{j=1}^m \lambda_{ij}^2}}{\lambda_{ij}} \right)$$

Finally, it uses the matrix P to rotate the original matrix X by two levels approximation. Level one, it calculates the matrix $L = (\Lambda'_R \Lambda_R)^{-1} \Lambda'_R P$. Then, it

4. New Unsupervised Feature Selection Technique for Noisy Data

normalizes the L by column to a transformation matrix $Q = LD$, where $D = 1/\sqrt{\text{diag}(L'L)}$ is the diagonal matrix that normalizes the columns of L . So, the preliminary rotated matrix is

$$f_{promax-temp} = Q^{-1}f_{varimax}$$

by reason of, $\text{Var}(f_{promax-temp}) = (Q'Q)^{-1}$ and the diagonal elements do not equal 1.

Level two, the rotated matrix is modified by matrix $C = \sqrt{\text{diag}((Q'Q)^{-1})}$ to $f_{promax} = Cf_{promax-temp}$ the rotated factor pattern is

$$\Lambda_{Promax} = \Lambda_R QC^{-1} \quad (4.2)$$

The coefficients in the rotated data is smaller, but the absolute distance between them significantly increased. It improves the quality of posterior analysis (i.e. clustering, classification).

4.4 RIFS Algorithm Description

In the this section, we will introduce our Robust Independent Feature Selection (RIFS) algorithm.

First of all, the independent components are computed from the X . Let S be a matrix whose rows are the independent decomposition vector of the matrix X and $V = [v_1, v_2, \dots, v_M]$, $v_i \in R^{M'}$ is the columns of S . Each vector v_i represents the projection of the i 'th feature (variable) of the vector X to the new dimensional space, that is, the M' elements of v_i correspond to the weights of the i 'th factor on each axis of the new subspace. The key observation is that features that are highly correlated or have high mutual information will have extremely similar weight (changing the sign has no statistical significance). On the two extreme sides, two independent features have maximally separated weight vectors; while two fully correlated features have identical similar absolute weights vectors.

Technically, the ICA method decomposes a multivariate data into independent latent sources and white noise is an underlying source that is also drawn out as an independent component by ICA. Let $S = [s_1, s_2, \dots, s_{m'}]$, $s_i \in R^M$ be the rows of S . The s_{unn} is representing white noise when the M elements of s_{unn} have much the same absolute value with finite variance, because the white noise is randomly having equal intensity at different features [197].

In order to isolate the noise, we use the Promax method to rotate the projected feature vectors v_i s to $RV = [rv_1, rv_2, \dots, rv_m]$, $rv_i \in R^{M'}$ with power k . It forces the structure of the factors loading to become bipolar that subsequently facilitates the

4. New Unsupervised Feature Selection Technique for Noisy Data

noise isolation from the main hidden sources. It quite mitigates the drawback of the noise during discriminative analysis by uniforming the factors load of s_{wn} .

To find the best subset, we look for the profoundly cross-correlated features subset by using the underlying factor structure of the RV_i and k_mean . The features of random vector X are clustered to $C = [c_1, \dots, c_d]$ when c_j represents $j'th$ cluster. We consider selecting d feature from M feature candidates.

In continue, the centroid of any cluster is computed:

$$C_j = \frac{1}{m_j} \sum_{rv_i \in c_j} rv_i \quad (4.3)$$

where m_j is the size of $j'th$ cluster.

Then, in any cluster the feature vectors rv_i are ranked based on their similarity with cluster centroid:

$$similarity(rv_i, C_j) = \frac{rv_i \cdot C_j}{\|rv_i\| \times \|C_j\|} \quad (4.4)$$

Where values range between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar.

We select the highest ranked rv_i for each cluster as a corresponding vector and the corresponding feature x_i is chosen as an independent representative feature. The selected features depute each cluster properly in terms of escalated spread, independence and restoration.

We summarize the complete RIFS algorithm for feature selection in Algorithm (1).

4.4.1 Computational Complexity Analysis

The computational cost for the main steps of our algorithm can be computed as follows:

- The ICA computational cost is $O(NM(1 + M)d')$ where M is the number of features/dimensions, N is the number of samples, and d' is the number of iterations in fastICA algorithm.
- The K-Means and Promax algorithms are utilized on just lower dimension including M points with $M' - dimensional$ vectors, so their computational costs are negligible.

Therefore, where $M' \ll N$ and d' is customarily fixed as a constant 200, the total computational cost of RIFS is roughly corresponding to the performance of fastICA. So the total cost of our RIFS algorithm is $O(NM(1 + M)d')$.

Algorithm 1 : RIFS for Feature Selection

Require: N data points with M features;
 $d < M$: the number of selected features ;
 k : the power of Promax rotation;

Ensure: d selected features

- 1: Compute the Independent Components as discussed in Section 3.1. Let $V = [v_1, v_2, \dots, v_M]$, $v_i \in R^{M'}$ contain feature decomposition vectors and M' is the number of hidden independent components.
 - 2: Rotate the V to RV as discussed in Section 3.2, with power coefficient set to k . We get $RV = [rv_1, rv_2, \dots, rv_M]$, $v_i \in R^{M'}$.
 - 3: Isolate the noise s_{wn} .
 - 4: Cluster the vectors rv_i to d categories $C = [c_1, \dots, c_d]$ by K-Means algorithm. Let C_j be the centroid of cluster c_j according to Eq.(3).
 - 5: Compute the **similarity** score for each feature vectors rv_i according to Eq.(4)
 - 6: Return the corresponding feature x_i of the most similar feature vector rv_i to the cluster's centroids for each d cluster.
-

4.5 Empirical Study

In this section, we have carried out several experiments to show the robustness, efficiency and effectiveness of our proposed RIFS method for unsupervised feature selection. The experiments consider both unsupervised (clustering) and supervised (classification) study. In the experiments, we have compared the RIFS, Laplacian Score and Maximum Variance. Laplacian Score and Maximum Variance are both state-of-the-art feature selection algorithms (filter methods), so this comparison makes possible to examine the efficacy of our proposed RIFS method.

4.5.1 Parameter Selection

Our RIFS has only one parameter, which is the k in performing the Promax rotation. We carried out different experiments in order to estimate the optimum value of k . RIFS achieves stable good performance with the k between 2 and 4 on all the four data sets. When k is less than 2, the performance slightly decreases as the k decreases. We assume $k = 4$ entire all experiments (both unsupervised and supervised study), in order to bring into uniformity.

4.5.2 Data Sets

We used four real world data sets in our experiments. The basic statistics of these data sets are outlined below in Table(4.1):

4. New Unsupervised Feature Selection Technique for Noisy Data

Table 4.1: Summary of four benchmark data sets

Data set	Instance	Feature	Classes
YALE	165	1024	15
ISOLET	1560	617	26
USPS	9298	256	10
COIL20	1440	1024	20

- The first one is **YALE**² face database which contains 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink. The original images are normalized (in scale and orientation) in order that the two eyes have been aligned at the same level. Then, we have cropped the face area into the final images for processing. The size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. Thus, each face image can be represented by a 1024-dimensional vector.
- The second one is **ISOLET**³ spoken letter recognition data. It contains 150 subjects who spoke the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1 through isolet5. In our experimentation, we use isolet1 which consists 1560 examples with 617 features.
- The third one is the **USPS**³ handwritten digit database. A famous subset contains 9298 16×16 hand written digit images in total.
- The fourth one is **COIL20**³ image library from Columbia which contains 20 objects. The images of each object were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. The size of each image is 32×32 pixels, with 256 gray levels per pixel.

4.5.3 Study of Unsupervised Cases

In this subsection, we apply our feature selection algorithm to clustering. The k-means clustering is performed by using the selected features subset and compare the results of both different algorithms and noise varieties.

²<http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html>

³<http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html>

4. New Unsupervised Feature Selection Technique for Noisy Data

4.5.3.1 Evaluation Metric

We evaluate the clustering result by informative overlapping between the obtained label of each data point using clustering algorithms and the label provided by the data set. We use the normalized mutual information metric (NMI)[186] as a performance measure. Let C indicate the set of clusters collected from the ground truth and C' obtained from a clustering algorithm. Their mutual information metric $MI(C, C')$ is defined as follows:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot \log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)} \quad (4.5)$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a data point arbitrarily selected from the data set belongs to the clusters c_i and c'_j , respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected data point belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information NMI as follows:

$$NMI(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))} \quad (4.6)$$

where $H(C)$ and $H(C')$ are the entropies of C and C' , respectively. It is easy to check that $NMI(C, C')$ ranges from 0 to 1. $NMI = 1$ if the two sets of clusters are identical, and $NMI = 0$ if the two sets are independent.

4.5.3.2 Clustering Results

In order to randomize the experiments, we evaluate the clustering performance with different number of clusters ($K=7, 11, 13, 15$ on YALE; $K=3, 5, 7, 10$ on USPS; $K=5, 10, 15, 20$ on COIL20 and $K=10, 15, 20, 26$ on ISOLET). For each given cluster number K (except using the entire data set), 10 tests were conducted on different randomly chosen clusters. Then, for each data set, the overall average performance

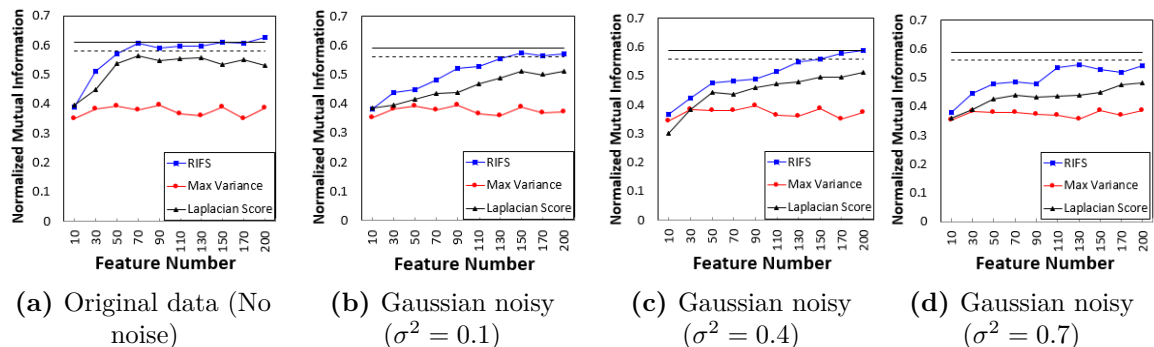


Figure 4.2: Clustering performance vs. the number of selected features on YALE.

4. New Unsupervised Feature Selection Technique for Noisy Data

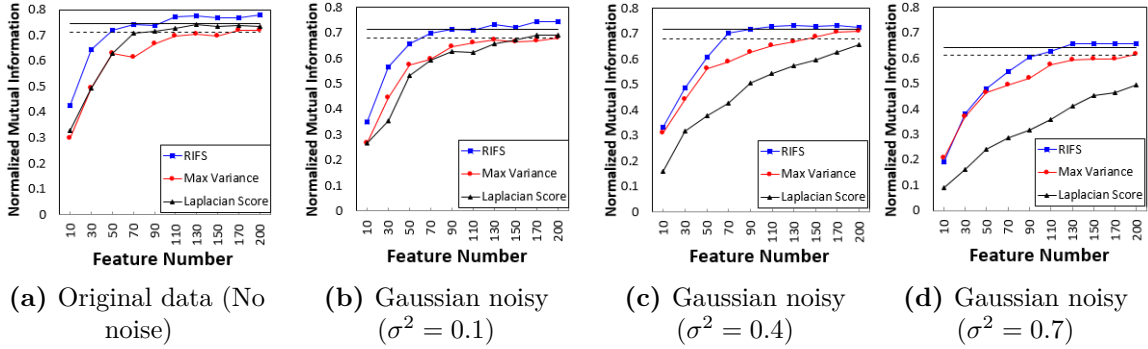


Figure 4.3: Clustering performance vs. the number of selected features on Isolet.

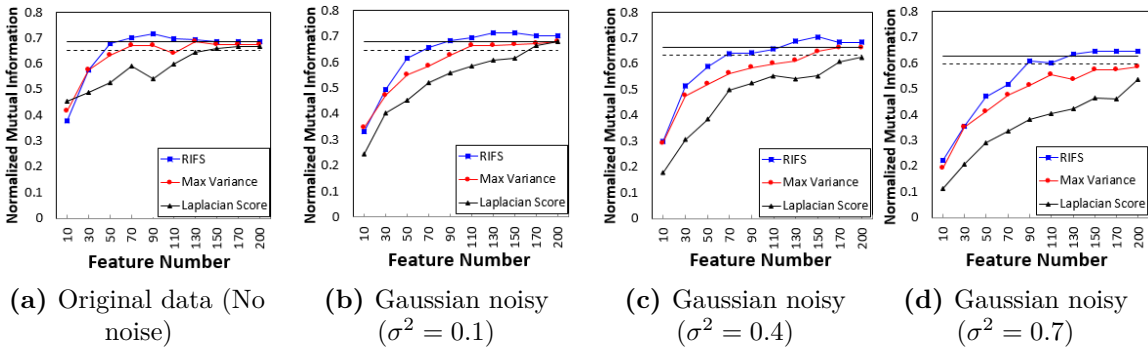


Figure 4.4: Clustering performance vs. the number of selected features on USPS.

as well as the standard deviation was computed over all tests with different cluster number K . In each test, we applied different algorithms to select d features and applied k-means for clustering. In order to initiate the k-mean starting point, we applied the Hierarchical Clustering algorithm [198] then the obtained d clusters centroids are used as k-mean starting points. In principal, we performed the above procedure on clean data sets. Then, we added different Gaussian noise ($\sigma^2 = 0.1, 0.4, 0.7$) to the original data sets and repeated the above clustering producer. For each σ^2 value, 10 random noise generated and tests executed, and both the average performance and standard deviation recorded over these 10 tests.

Fig.(4.2,4.3,4.4 and 4.5) present the plots of clustering performance versus the

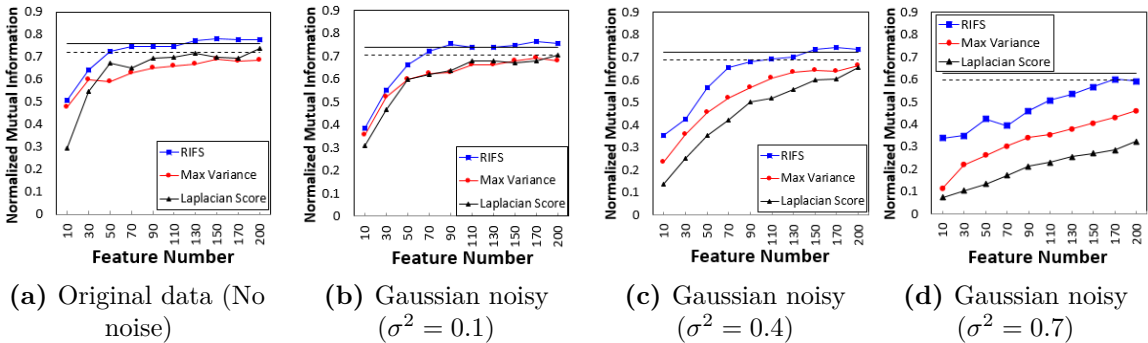


Figure 4.5: Clustering performance vs. the number of selected features on COIL20.

4. New Unsupervised Feature Selection Technique for Noisy Data

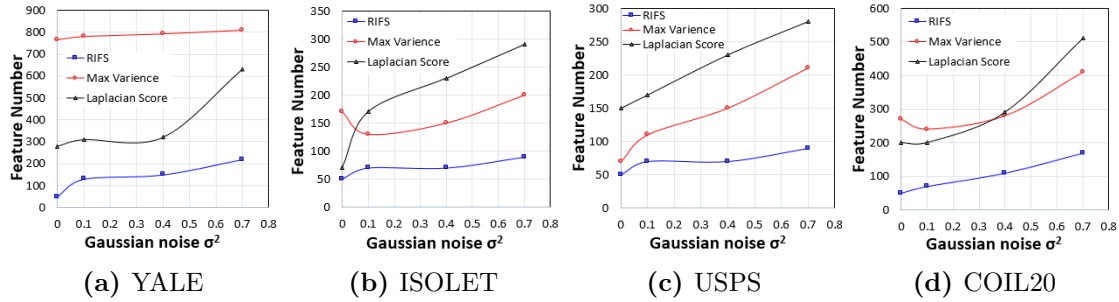


Figure 4.6: The noise level vs. the number of selected feature that is needed to achieve the 95% of clustering performance with all features.

Table 4.2: The proportion of features ($\#$ selected features/ $\#$ all features%) that is needed to achieve the 95% of clustering performance with all features.

	Method	No noise	$\sigma^2 = 0.1$	$\sigma^2 = 0.4$	$\sigma^2 = 0.7$	Average
YALE	RIFS	4.9 \pm 3.8	12.7 \pm 3.2	14.6 \pm 3.7	21.5 \pm 2.5	13.4
	Laplacian Score	27.3 \pm 2.3	30.3 \pm 3.9	31.3 \pm 5.1	61.5 \pm 8.6	37.6
	Max Variance	74.7 \pm 1.1	76.2 \pm 3.2	77.4 \pm 3.1	79.1 \pm 1.4	76.9
ISOLET	RIFS	8.1 \pm 2.2	11.3 \pm 3.1	11.3 \pm 4.3	14.6 \pm 4.4	11.3
	Laplacian Score	11.3 \pm 3.6	27.6 \pm 11.2	37.3 \pm 7.8	47.0 \pm 9.0	30.8
	Max Variance	27.6 \pm 8.1	21.1 \pm 6.2	24.3 \pm 4.3	32.4 \pm 11.1	26.3
USPS	RIFS	19.5 \pm 1.7	27.3 \pm 2.3	27.3 \pm 1.8	35.2 \pm 4.6	27.3
	Laplacian Score	58.6 \pm 8.1	66.4 \pm 17.2	89.8 \pm 9.8	100.0 \pm 0.0	78.7
	Max Variance	27.3 \pm 5.2	43.0 \pm 7.9	58.6 \pm 15.1	82.0 \pm 12.0	52.7
COIL20	RIFS	4.9 \pm 3.3	6.8 \pm 2.3	10.7 \pm 5.1	16.6 \pm 7.7	9.8
	Laplacian Score	19.5 \pm 8.8	19.5 \pm 6.3	28.3 \pm 9.1	50.0 \pm 12.8	29.3
	Max Variance	26.4 \pm 5.4	23.4 \pm 6.2	27.3 \pm 8.7	40.0 \pm 3.2	29.3

number of selected features d on ISOLET, USPS, COIL20 and YALE, successively, without and with different level of Gaussian noises. As shown in the plots, our proposed RIFS algorithm persistently surpasses both competitors on all the four data sets and noise levels. From the plot (a) of each Fig.2 ~ 5 (noise less), we can see RIFS converges to the best result in double quick time, with approximately 50 features. Meanwhile, both other methods mostly require more than 100 features (in average) to achieve 95% of the best result. When we add Gaussian noise with higher standard variance, we need to select more features to achieve reasonable clustering performance, as it can be seen in the plot (b ~ c) of each Fig.2 ~ 5. However, in RIFS case, this trend is very slightly pronounced when the performance of the other methods is reduced quickly by increasing the Gaussian noise standard variance, as it can be seen in Fig.(4.6). It would be worth mentioning that, on the ISOLET data set, our proposed RIFS algorithm performs strangely robust against the noise by selecting few more features. For example, in $\sigma^2 = 0.4$ case only 70 features are selected by RIFS and the clustering normalized mutual information is 70.3%, which is almost equal to the clustering result by using all the 617 features (71.7%). However, the

4. New Unsupervised Feature Selection Technique for Noisy Data

Max Variance and Laplacian Score perform comparably to one another on original ISOLET data set but the Laplacian Score shows higher sensitivity to the noisy data. On COIL20 data set the Max Variance and Laplacian Score perform comparably to one another while Max Variance becomes obviously better than Laplacian Score On USPS data set. On YALE data set, Laplacian Score completely performs better than Max Variance, roughly, Max Variance does not have any function on YALE data set, possibly, due to the fact that sample size is small. The most surprising aspect of the result is that Max Variance slightly performs worse on original than data with light noise ($\sigma^2 = 0.1$) on COIL2 and ISOLET data sets.

The main objective of our experiment is to reduce the dimensionality of the data by taking to account the robustness against the noise, in Table (4.2), we report the selected feature proportion for achieving to at least 95% of the best clustering performance by using all features for each algorithm and Gaussian noise standard variance. The last column of each table records the average selected feature proportion over different standard variance of Gaussian noise. As it can be seen, RIFS significantly outperforms both other methods on all the four data sets. Laplacian Score performs the second best on YALE data set. Max Variance performs the second best on USPS and ISOLET data sets. Max Variance and Laplacian Score perform comparably to one another on COIL20 data set. Comparing with the second best method, RIFS selects 24.2%, 15.0%, 25.4% and 19.5% less proportion of features in average for reaching to the at least 95% of clustering performance with all features, when measured by normalized mutual information on the YALE, ISOLET, USPS and COIL20 data sets, respectively.

4.5.4 Study of Supervised Cases

In this experiment, we examine the discriminating capability of the different feature selection methods. The 1-Nearest Neighbor (1NN) classifier is used and we assume that well-selected feature subset should yield more accurate classifier [191]. We perform leave-one-out cross validation as follows: For each data point x_i , we find

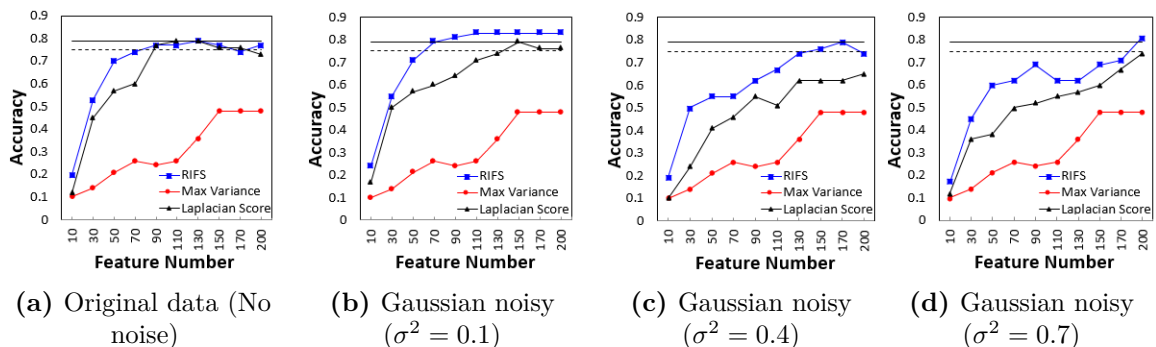


Figure 4.7: Classification accuracy vs. the number of selected features on YALE.

4. New Unsupervised Feature Selection Technique for Noisy Data

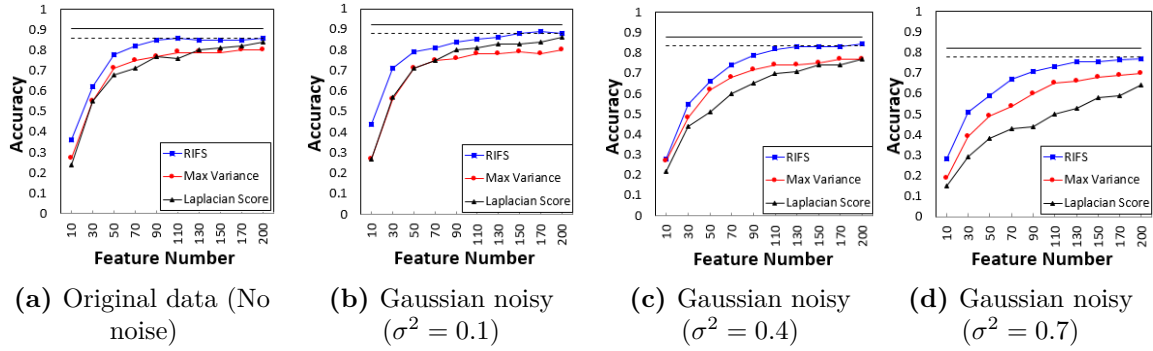


Figure 4.8: Classification accuracy vs. the number of selected features on Isolet.

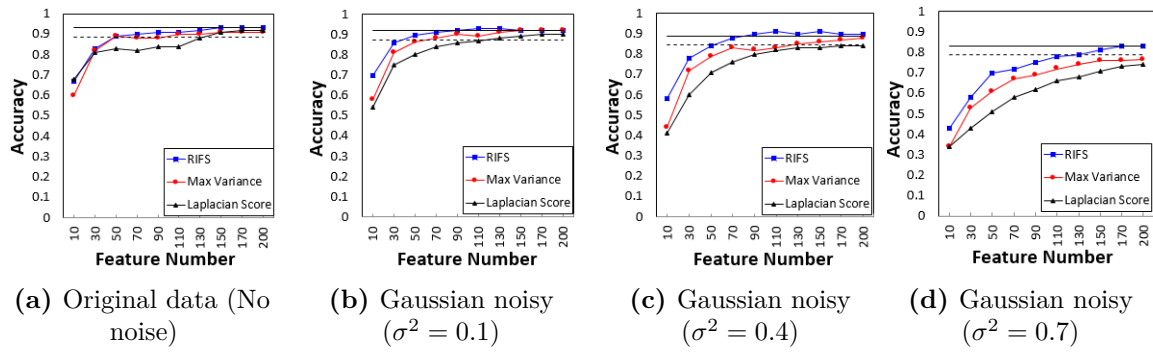


Figure 4.9: Classification accuracy vs. the number of selected features on USPS.

its nearest neighbor x'_i . Let $c(x_i)$ be the class label of x_i . The nearest neighbor classification accuracy rate (AR) is thus defined as

$$AR = \frac{1}{N} \sum_{i=1}^N \delta(c(x_i), c(x'_i)) \quad (4.7)$$

where N is the number of data points and $\delta(a, b) = 1$ if $a = b$ and 0 otherwise. All results reported in this section are obtained by averaging the accuracy from 10 trials of experiments. Fig. (7 ~ 10) represent the plots of 1-nearest neighbor classification accuracy rate versus the number of selected features. As it can be seen, roughly on all the four data sets, RIFS at every turn goes one better than both other methods.

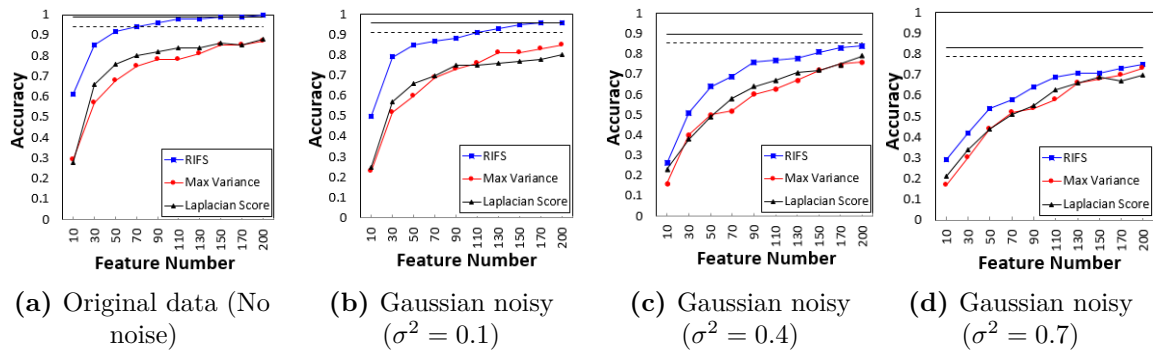


Figure 4.10: Classification accuracy vs. the number of selected features on COIL20.

4. New Unsupervised Feature Selection Technique for Noisy Data

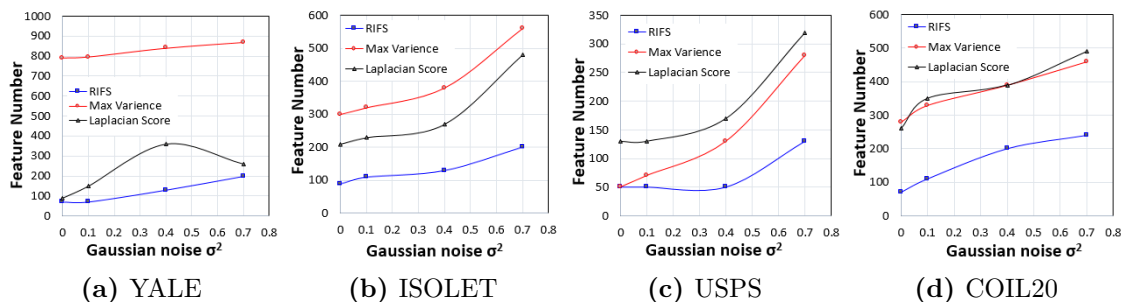


Figure 4.11: The noise level vs. the number of selected feature that is needed to achieve the 95% of classification accuracy with all features.

Table 4.3: The average proportion of features ($\#$ selected features/ $\#$ all features%) that is needed to achieve the 95% of classification accuracy rate with all features.

	YALE	ISOLET	USPS	COIL20
RIFS	11.7	21.5	27.3	15.1
Laplacian Score	21.0	48.2	67.4	36.4
Max Variance	80.4	63.2	51.8	35.6

Almost identical to clustering or even better, RIFS converges to the best result quickly on original (No noise) data set, with less than 90 features (in average) and it shows strange robustness against the noises. For example in case of $\sigma^2 = 0.7$, RIFS selects approximately 100 more features in average to converge to the best result on all data sets, as it can be seen in Fig.(11). Remarkably, on the USPS data set with moderate additive Gaussian noise ($\sigma^2 \leq 0.4$), RIFS consistently can achieve 95% of the best classification accuracy by using no more than 50 features. On this data set, the Max Variance algorithm performs comparably to our algorithms and much better than Laplacian Score, and Laplacian Score's performance is defected more by additive noise. On the COIL20 data set, the Laplacian Score and Max Variance algorithms perform comparably to each other. On the ISOLET data set, the Laplacian Score and Max Variance algorithms perform comparably to each other, and Laplacian Score performs the worst. Surprisingly, on the YALE data sets, Max Variance algorithm performs quite bad in both with and without noise, unless selecting approximately 80% of features and Laplacian Score performs better on noisy data with Gaussian noise $\sigma^2 = 0.7$ than $\sigma^2 = 0.4$, possibly, due to the fact that sample size is small. The same as unsupervised study, in Table (3), we report the average (over all Gaussian noise $\sigma^2 \in \{0, 0.1, 0.4, 0.7\}$) selected feature proportion for achieving to at least 95% of the best classification performance by using all features for each algorithm. As it can be seen, RIFS achieves to the 95% of the best classification performance with approximately two times less numbers of features than the second best competitor on all data sets.

5

New Method to Homogenize the Density

Contents

5.1	Introduction	78
5.2	Related work	80
5.3	The problem of multi-density	82
5.4	Background	83
5.4.1	Self-organizing map	84
5.4.2	Multilinear Transformation	85
5.5	Feature Space Curvature Map	87
5.5.1	Feature Space Curvature Modeling	88
5.5.2	Curvature Map	90
5.6	Application of FSCM in the Real Data	93
5.6.1	Datasets	93
5.6.2	Evaluation Metric	94
5.6.3	Experiment Setup	95
5.6.4	Clustering Results	97
5.6.5	Complexity Analysis	98

In this chapter, we introduce our Feature Space Curvature Map method that we propose to homogenize the cluster density which is important to enhance the density-based clustering algorithm to categorize the varied density data.

5.1 Introduction

Density-based clustering algorithms are now widely used in a variety of applications, ranging from agriculture [199], high energy physics [200], material sciences [201], social network analysis [202] to molecular biology [203]. These approaches regard clusters

5. New Method to Homogenize the Density

as regions in the feature space in which the data points are dense and separated by regions of low data point density (noise). However, they fail to properly find clusters in data exposing different densities in various regions of the feature space. This failure results from using a single global density threshold on all the data points.

We refer to Multi-density or varied densities clusters as the clusters in different regions of the feature space that are formed in considerably different densities [204]. Typically, density-based clustering algorithms require the user to specify the parameters (typically one or few constants) that define the density-level thresholds, as an input to the algorithm. Properly selecting the value of those parameters becomes even more difficult when the data exposes a multi-density structure since a single density threshold parameters that appropriately detect such structure would be different in various regions of the feature space. Therefore, our main goal is to introduce a new preprocessing method for homogenizing the density of data in order to enable existing single density threshold algorithms to properly identify the actual clusters structure throughout the whole feature space.

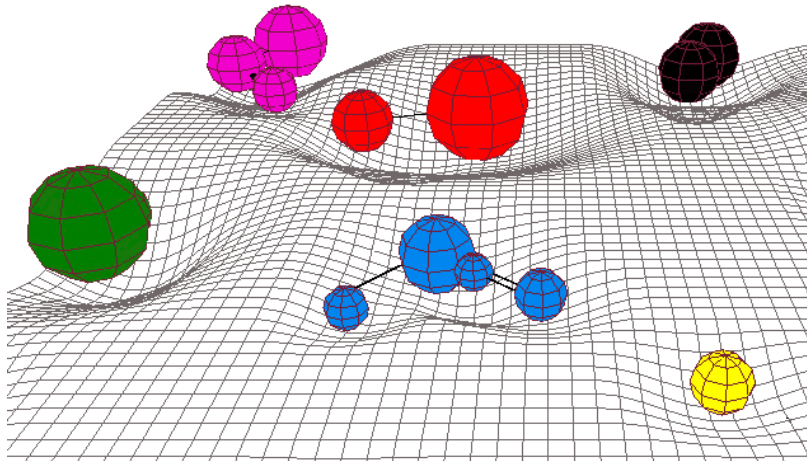


Figure 5.1: The stars (distinguished by colors) bend space-time (grid), and the gravitational force among the stars (black lines) can be described by space-time curvature.

In analogy to the Theory of Relativity [205], the computation of distance between objects changes when objects placed in a space introduce local curvatures in it. In this work, we propose a method to map an originally euclidean feature space into a non-euclidean one with local curvatures introduced by the presence of the data points themselves. Projecting data points to their new coordinates in this transformed space, we observe a more uniform density distribution and we show how traditional clustering algorithms in this projected dataset result in significantly better capacity to identify its structure. The key idea is that data clouds can bend the feature space based on their density, which represents the density structure of the data.

5. New Method to Homogenize the Density

According to the Theory of Relativity, see Fig.5.1, the planets warp space-time, and the standard quantized version of the theory includes massless gravitons delivering the gravitational force. The space-time curvature is a quantity describing how the local geometry of a subspace differs from the flat space around any single planet. In our method, we assume that a data cluster bends the m -dimensional feature space locally based on their shape and density; and we use this properties as a source to model the data density structure.

In particular, our approach includes two main steps: first, we apply our new Gravitational Self-Organizing Map (GSOM) method to compute the data density structure by using the gravity force in terms of relationships between clusters rather than distance. It computes the feature space bending with a correct topology preserving map to present the potentially complex multi-density structure of the data. Second, we apply our novel parametric multilinear transformation, using the GSOM map, to project the data points to a new linearized and euclidean feature space with more uniform density distribution. Any existing density-based clustering algorithm can then be applied to the projected data to identify its clusters. This very often leads to better clustering analysis performance than using original data, without applying algorithmic changes to the clustering algorithm itself.

Note that the multilinear transformation is a spatial transformation which has commonly been used in Mathematical Physics [206] and Graphics for video compression [207]. The novelty of our approach is that we apply multilinear transformation to map an originally euclidean feature space into a new euclidean space by a given curvature model, in our case computed by the GSOM method; where the projected data points show more uniform density distribution.

The rest of this chapter is organized as follows: The intuition and motivation of the multi-density clustering analysis is presented along with an overview of related work and methods in section 5.2. In section 5.3, the basic notions and the problem of multi-density are defined. In section 5.4, the background techniques are briefly reviewed. In section 5.5, our new Feature Space Curvature Map (FSCM) algorithm is described. The experimental result of our FSCM method is illustrated in section 5.6.

5.2 Related work

Density-Based Clustering refers to the methods that detect clusters in the data based on the idea that a cluster in a feature space is an adjacent region of concentrated points, separated from other clusters by regions that are empty or sparse. DBSCAN [208] computes the density of each data point by counting the existing data points in its *eps*-neighbourhood. However, using a single *eps* can often not adequately

5. New Method to Homogenize the Density

characterize the datasets with clusters of very different densities. Several approaches have been proposed to cope with this weakness.

Many efforts have been devoted to solving the varied densities problem by variants of DBSCAN. The HDBSCAN($\hat{\epsilon}$) [209] and OPTICS [11] address this issue by producing the reachability plot to extract clusters. The reachability plot is a 2D plot, with the ordering of the points as processed by these algorithms on the x – axis and the reachability distance on the y – axis. As points associating to a cluster have a low reachability distance to their nearest neighbor, the clusters appear as valleys in the reachability plot. However, they are methods to visualize the cluster structures without producing a clustering result explicitly. Thus, these methods manually need to extract the clusters, with varying densities, from the reachability plot. Our approach mathematically combines the strengths of statistical and topological methods to eliminate the need for expert human visual analysis.

The VDBSCAN [210] and MSDBSCAN [211] partition a dataset into different density level sets by statistical analytic, and then estimate eps for each density level set, finally use DBSCAN clustering on each density level set with corresponding eps and $minPts$ to get clustering results. Although, they are not suitable for large datasets since they require several passes of the data and consume high computational time. Our method is more efficient since it is a preprocessing step which enables us to carry out a single pass clustering. Moreover, it is scalable approach as it uses the SOM-based iterative resampling schemes rather the whole dataset.

Recently, neighborhood-based density estimator approaches use local density estimation to overcome the issues of varying densities. DP [212] instead of finding core points uses a global threshold in the first step, it finds the density peak of every cluster and then links the neighboring points of each peak to form a cluster. LC-CFSFDP [213] and DPC-DBFN [214] improve the clustering performance of CFSFDP by enhancing local density estimators based on a kNN graph and a fuzzy neighbourhood measure, respectively. However, most of these algorithms face problems in handling large datasets since they require high storage to keep the distance matrix. In contrast, we use the GSOM to efficiently learn and build a density structure map of data by preserving its neighboring relations since this structure requires significantly lower storage than the distance matrix.

More recently, Density-ratio based scaling methods have also been applied to handle the multi-density data. The density-ratio of a point is the ratio of two density estimates which are calculated by using the same density estimator, but with two different bandwidths. ReScale [215] enables a density-based clustering algorithm to identify clusters with varied densities by rescaling the given dataset. Nevertheless,

5. New Method to Homogenize the Density

it rescales each individual feature independently and if a significant overlap exists between clusters on some features, ReScale may become less effective.

Rather than rescaling on each individual feature, [216] proposes a new density-ratio DScale method which rescales the pairwise distance as a multi-dimensional scaling, such that points located at locally high-density areas have higher densities than points located at locally low-density areas. However, DScale fails to be a globally valid CDF (Cumulative Distribution Function) transformation for the entire dataset since it is valid within the λ -neighbourhood of each data point which is treated independently.

Furthermore, CDF-TS [217] density-ratio method applies the same CDF transform process as DScale with an additional “shift” to ensure that: (i) the transformed-and-shifted dataset becomes approximately uniformly distributed in the scaled λ -neighbourhood; and (ii) then we can use standard Euclidean distance to measure distances between any two transformed-and-shifted points. These advantages are not available in DScale which relies on a rescaled distance which is non-metric and asymmetric. However, CDF-TS does not preserve potentially complex topological structure of clusters since it still uses the CDF scaling. In contrast, our method uses a novel parametric and global mapping function that preserves potentially complex topological density structure.

The intention of our work is to present a novel generalized technique to homogenize the density of adjacent clusters of varying density. Our method introduces analogies between the data analysis domain and the Theory of Relativity in physics by using the tensor calculus and a modified Self-Organizing Map method; Thereby, our FSCM is a parametric, scalable, automated method and it does not depend on any particular clustering algorithm. Especially, in our GSOM we use the SOM-based iterative resampling schemes to optimize the computation cost and increase the efficiency and scalability.

5.3 The problem of multi-density

In this section, we firstly provide a brief mathematical notation to use throughout this chapter, then formalize the general weakness of existing density-based clustering algorithms to stratify multi-density clusters.

We use X to indicate a dataset of n data points $X = (x_1, x_2, \dots, x_n)$ where $x_i \in R^m$. Let $pdf(x)$ and $\widehat{pdf}(x)$ stand for the true density of point x and its estimation based on sample data respectively. Besides that, let $\mathcal{N}(x, \epsilon) = \{x' \in X | d(x, x') \leq \epsilon\}$ denote the ϵ -neighbourhood of x , where $d : R^m \times R^m \rightarrow R$ is the euclidean distance function.

5. New Method to Homogenize the Density

Technically, the density-based clustering algorithms (e.g. DBSCAN [208]) use a small ϵ -neighbourhood to estimate the $pdf(x)$ as follows:

$$\widehat{pdf}(x, \epsilon) = \frac{|\mathcal{N}(x, \epsilon)|}{n\mathcal{V}(\epsilon)} \quad (5.1)$$

where $\mathcal{V}(\epsilon)$ is the volume of an m -sphere of radius ϵ [216].

Let $C = \{c_1, c_2, \dots, c_q\}$ denote a set of non-overlapping and non-empty clusters where $c_i \subset X$, $\forall_{i \neq j} (c_i \cap c_j = \emptyset)$ and $c_i \neq \emptyset$. Let $m_i = \operatorname{argmax}_{x \in c_i} \widehat{pdf}(x)$ and $p_i = \widehat{pdf}(m_i)$ denote the data point with the mode (highest density) of cluster c_i and its corresponding peak density value respectively.

In [215], they present a condition to guarantee these density-based clustering algorithms can identify all clusters in a dataset. The condition is that the estimated density p_i at the mode m_i of each cluster is greater than the maximum of the minimum estimated density along any path through \widehat{pdf} which is linking any two modes:

$$\min_{i \in \{1 \dots q\}} p_i > \max_{j \neq k \in \{1 \dots q\}} g_{jk} \quad (5.2)$$

Where g_{jk} is the highest of the minimum density along the path that links clusters c_j and c_k .

This condition implies that a single density threshold τ must exist to fracture all trajectories between the peaks by nominating the regions with density less than τ to noise. Nevertheless, these density-based clustering algorithms fail to detect all clusters when the peak density of some clusters is lower than a low-density region between some other clusters.

In this chapter, we propose a parametric transformation which acts as a preprocessing step to tackle this issue by projecting a given dataset X to X' that has a more uniform density distribution and it also better fulfills this property. As a result, it enables clusters with varied densities in the original space to be identified using a single threshold in the transformed space, something that would be impossible in the original space.

5.4 Background

In this section, we discuss an overview of two techniques that we leverage and extend in our analytic approach. We firstly revisit the self-organizing map algorithm, then we review the multilinear transformation technique.

5.4.1 Self-organizing map

Kohonen's Self-Organizing Map (SOM) [218] is one of the popular types of neural network which preserves and retains an accurate representation of the topology of the data space.

The SOM often arranges a set of neurons in a 2-D rectangular or hexagonal grid \mathbb{T} in size ς , to establish a discrete topological mapping of an input space $X \in R^{n \times m}$. Ω is the set of neuron indexes. The neurons are represented by a set of weight vectors $V = \{v_1, v_2, \dots, v_\varsigma\}$, where v_i is the weight vector associated with neuron i and is a vector of the same dimension $-m-$ of the input, ς is the total number of neurons, and let r_i be the location vector of neuron i on the grid. At the start of the learning, all the weights are initialized to small random numbers. Then the algorithm repeats next two steps until the map converges in order to preserve maximum topological properties of the data on the map.

Each iteration t , it first chooses one random point $x(t)$ of the dataset and selects the winner neuron:

$$\nu(t) = \underset{k \in \Omega}{\operatorname{argmin}} \|x(t) - v_k(t)\| \quad (5.3)$$

Then it updates the weights of the winner and its neighbors:

$$\Delta v_k(t) = \alpha(t) \eta(\nu(t), k, t) [x(t) - v_{\nu(t)}(t)] \quad (5.4)$$

where the coefficient $\alpha(t)$ is termed the 'learning rate' which is scalar-valued and it decreases monotonically [218]:

$$\alpha(t) = \alpha_0 \cdot e^{\frac{-t}{\lambda_\alpha}} \quad (5.5)$$

η is the neighborhood function which quantifies the propagation and decay of the updates to the winner node on its neighbours through the grid topology. The Gaussian form is often used in practice, specifically:

$$\eta(\nu(t), k, t) = \exp \left[-\frac{\|r_{\nu(t)} - r_k\|^2}{2\sigma(t)^2} \right] \quad (5.6)$$

Where $\sigma(t)$ represents the effective range of the neighborhood radius around $\nu(t)$. Like the learning rate, the neighborhood similarly decays with an exponential decay function:

$$\sigma(t) = \sigma_0 \cdot e^{\frac{-t}{\lambda_\sigma}} \quad (5.7)$$

where λ_α and λ_σ are the decay time constants.

5. New Method to Homogenize the Density

Note that the initial learning rate α_0 , radius σ_0 and both time constants are empirically chosen based on the application. In our work, we later present their value in the section 5.5.

The "No Move" [15] criteria is widely used to detect convergence of the learning mechanism. It considers stopping condition that defines no-improvement in SOM's status as no training samples change their best match unit in a complete iteration of the training set. However, when we use "No Move" criterion, then there could be a case where the weights kept oscillating between iterations, thus causing the criterion is never met. Therefore, a iteration threshold is necessary to be selected whether the SOM doesn't converge.

Furthermore, "Mean Distance to the Closest Unit" (MDCU) criteria is the quantization error of the SOM map [218] which is computed after the training process. It calculates the average distance of the sample vectors to the node centroids by which they are represented.

Finally, the SOM provides a topology preserving map [202] from input to output spaces, which includes grid \mathbb{T} and weight vectors V . For SOM training, the weight vector associated with each neuron moves to become the center of a local group of input vectors. The group i is represented by its centroid vector v_i and the local groups are connected via \mathbb{T} .

In this work, we enhance the generic SOM technique to model the feature space curvature by plugging the gravitation and fabric of space concepts into the standard SOM, in analogy with Relativity theory, to model the density structure of dataset and still describe the whole original feature space.

5.4.2 Multilinear Transformation

The multilinear transformation is a spatial transformation function which is locally linear but the coefficients change across different regions of the space. In general, a multilinear transformation function $\mathcal{M} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ of m variables is called a m -linear map. To represent it visually, we describe in details the bilinear transformation function $\mathcal{B} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, which is the multilinear map of two variables [206].

In Fig.5.2, on the 2D Cartesian coordinate system, we have a quadrilateral (left image) that we want to transform into a rectangle (right image). To interpolate each point ρ on the arbitrary quad into ρ' on the rectangle, we need to obtain a bilinear map function which describes the entire point space enclosed by the quadrilateral [207].

We need to compute the function which transfers the quadrilateral into the rectangle. We assume there are bilinear mapping functions \mathcal{B}_x and \mathcal{B}_y :

$$\begin{aligned} x &= a_1\mathcal{B}_x(x, y) + a_2\mathcal{B}_y(x, y) + a_3\mathcal{B}_x(x, y)\mathcal{B}_y(x, y) + a_4 \\ y &= a_5\mathcal{B}_x(x, y) + a_6\mathcal{B}_y(x, y) + a_7\mathcal{B}_x(x, y)\mathcal{B}_y(x, y) + a_8 \end{aligned} \quad (5.8)$$

5. New Method to Homogenize the Density

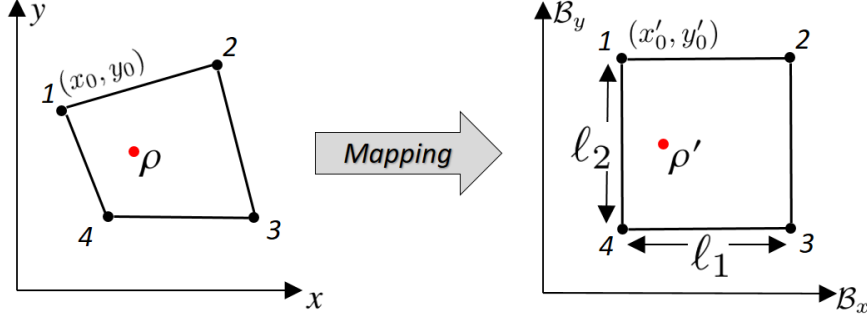


Figure 5.2: A bilinear transformation enables us to represent the arbitrary shaped quadrilateral as a rectangle.

Where $a_1 \dots a_8$ are the transformation parameters. When we have the values of \mathcal{B}_x and \mathcal{B}_y for four lateral points of the rectangle, we can compute the parameters by solving two linear systems, each of four equations with four unknowns.

Let $\vec{d}_1, \vec{d}_2, \vec{d}_3, \vec{d}_4$ be the displacement vectors of the rectangle corners which upper left one is point (x'_0, y'_0) and $\vec{d}_i = (d_i^x, d_i^y) = (x_i - x'_i, y_i - y'_i)$. We compute the parameters $a_1 \dots a_8$ of the bilinear transformation for these displacements as follows:

$$\begin{aligned}
 a_1 &= [(d_2^x - d_1^x)(\ell_2 - y'_0) + y'_0(d_4^x - d_3^x) + \ell_1 \ell_2] / \ell_1 \ell_2 \\
 a_2 &= [(d_1^x - d_3^x)(\ell_1 + x'_0) + x'_0(d_4^x - d_2^x)] / \ell_1 \ell_2 \\
 a_3 &= [d_2^x - d_1^x + d_3^x - d_4^x] / \ell_1 \ell_2 \\
 a_4 &= x'_0 + d_1^x - a_1 x'_0 - a_2 y'_0 - a_3 x'_0 y'_0 \\
 a_5 &= [(d_2^y - d_1^y)(\ell_2 - y'_0) + y'_0(d_4^y - d_3^y)] / \ell_1 \ell_2 \\
 a_6 &= [(d_1^y - d_3^y)(\ell_1 + x'_0) + x'_0(d_4^y - d_2^y) + \ell_1 \ell_2] / \ell_1 \ell_2 \\
 a_7 &= [d_3^y - d_1^y + d_2^y - d_4^y] / \ell_1 \ell_2 \\
 a_8 &= y'_0 + d_1^y - a_5 x'_0 - a_6 y'_0 - a_7 x'_0 y'_0
 \end{aligned} \tag{5.9}$$

To obtain the \mathcal{B}_x and \mathcal{B}_y , we solve the Equation system 5.8 to represent the \mathcal{B}_x and \mathcal{B}_y as a function of x and y . These functions are calculated as follows: we firstly solve for $\mathcal{B}_x(x, y)$ from the first equation of the system:

$$\mathcal{B}_x(x, y) = \left(\frac{x - a_4 - a_2 \mathcal{B}_y(x, y)}{a_1 + a_3 \mathcal{B}_y(x, y)} \right) \tag{5.10}$$

Then, substituting this into the second Equation of system 5.8, rationalizing the denominators and combining like powers of $\mathcal{B}_y(x, y)$, we find the following quadratic equation that must be solved to get $\mathcal{B}_y(x, y)$:

$$A \mathcal{B}_y(x, y)^2 + B \mathcal{B}_y(x, y) + C = 0 \tag{5.11}$$

5. New Method to Homogenize the Density

Where:

$$\begin{aligned} A &= a_6 a_3 - a_7 a_2 \\ C &= a_8 a_1 - a_5 a_4 + a_5 x - a_1 y \\ B &= a_8 a_3 - a_7 a_4 + a_6 a_1 - a_5 a_2 + a_7 x - a_3 y \end{aligned} \quad (5.12)$$

Finally, we choose the positive root of the quadratic Equation 5.11 for $\mathcal{B}_y(x, y)$ as a feasible solution.

As a result, we obtain a bilinear map function which describes the entire point space enclosed by the quadrilateral and lets us displace each point ρ on the arbitrary quad into ρ' on the rectangle continually. This transformation is likewise generalizable to the multi-dimensional space, which is called multilinear transformation to map a hyper-quadrilateral to a hyper-rectangle. Note that the equation system is no longer linear; However, it can be solved quite easily by applying analytical solutions such as Grobner bases [219].

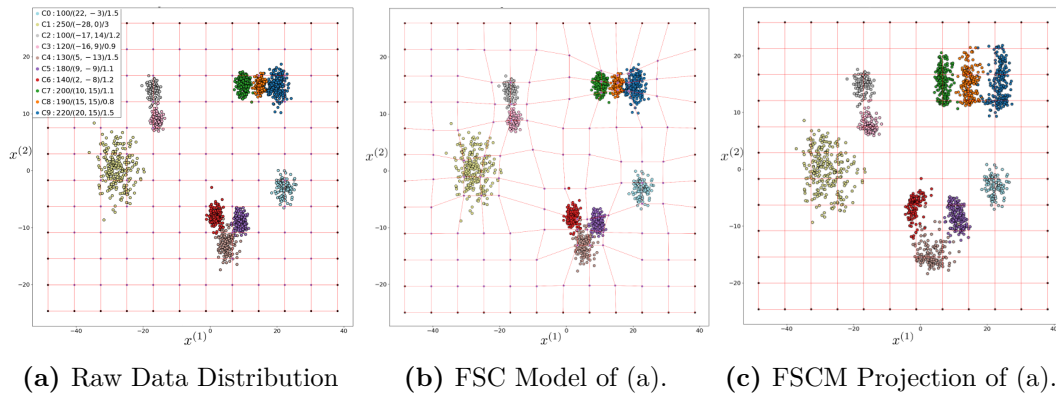


Figure 5.3: Application of FSCM on a multi-density 2D dataset Synt10 containing ten clusters. (a) A scatter plot of clusters with varied densities. The legend shows the $size/\mu(x^{(1)}, y^{(2)})/\sigma$ per cluster, the colors represent the data original labeling and the red lines draw the initial FSF. (b) shows the FSC model that is computed with our FSCM method. Note that the red lines show the deformation of the FSF. (c) scatter plots the data (a) projected by applying our transformation through model (b). As a result, the diversity of the clusters' density scaled appropriately to achieve a better density-based clustering performance.

5.5 Feature Space Curvature Map

Our Feature Space Curvature Map (FSCM) method involves two main steps that will be formally described in this section. First, we apply our new Gravitational Self-Organizing Map (GSOM) method to compute the clusters density structure. It computes the feature space bending with a correct topology preserving map to present the potentially complex multi-density structure of the data. Second, we apply our enhanced multilinear transformation, from the GSOM map to project the data points

5. New Method to Homogenize the Density

to the new euclidean feature space with more uniform density distribution. Note that the original feature space is typically considered to be euclidean that classic distances are defined on it. These steps perform a globally nonlinear transformation of the dataset to which any existing density-based clustering algorithm based on euclidean distances can be applied.

5.5.1 Feature Space Curvature Modeling

In the first step, we use a topological m -dimensional elastic mesh \mathbb{T} to construct the fabric of feature space as a smooth manifold with a Riemannian metric, while covering the whole feature space. We call this mesh the Feature Space Fabric (FSF) through this thesis. Then, we compute the Feature Space Curvature (FSC) model where the concentrations of data points can bend the FSF dramatically while the areas with less concentration just slightly bend it or leave it as locally euclidean. The FSC presents the density structure of the data.

In order to be able to derive the FSC, we propose a new Gravitational Self-Organizing Map (GSOM) algorithm. We train a GSOM network to learn the density topology of the input data X , while still covering the whole feature space. To plugging the concept of the gravity and fabric of space to the generic SOM, we apply four key enhancements on it.

5.5.1.1 Initialization Method

For initializing the neurons in topological space \mathbb{T} instead of random initialization, we use an m -dimensional Regular Rectangular Grid (RRG). Let I_j be the grid interval of the feature $x^{(j)} \in R^n$:

$$I_j = [\min(x^{(j)}) - h, \max(x^{(j)}) + h] \quad (5.13)$$

where h is the marginal coefficient to create extra space around the grid where $h = 0.15$ works well. Let ξ be the number of grid lines for each grid interval I_j where $\xi = \sqrt[m]{\xi}$ and $\xi \in \mathbb{Z}^{3+}$. This RRG slices the entire feature space into subdivisions to accurately capture and represent potentially complex local density structures of the data. See Fig.5.3a, the red lines represent an initial 2D RRG.

5.5.1.2 Rigid Boundary

We modify the SOM algorithm to keep the boundary nodes rigid, during the weight updating of the winner node and its neighbors, to avoid the crumpling and folding the RRG. The node ω is boundary if it belongs to:

$$s = \{\omega \in \Omega | \exists j (v_{\omega}^{(j)} = \min(I_j) \vee v_{\omega}^{(j)} = \max(I_j))\} \quad (5.14)$$

5. New Method to Homogenize the Density

where $j \in [1, \dots, m]$ and $s \subset \Omega$. To keep rigid the boundary nodes, we replace the Equation 5.4 with:

$$\Delta v_k(t) = \begin{cases} 0 & \text{if } k \in s \\ \alpha(t)\eta(\nu, k, t)[x(t) - v_\nu(t)] & \text{else} \end{cases} \quad (5.15)$$

See Fig.5.3a, the black nodes at the sides of RRG represent boundary ones which remain rigid after training, see Fig.5.3b.

5.5.1.3 Update Procedure

The Law of Universal Gravitation [205] states that every particle of matter in the universe attracts every other particle with a force that is directly proportional to the product of the masses of the particles and inversely proportional to the square of the distance between them. In like manner, we propose a novel Gaussian neighborhood function based on an assumed mutual force acting between all pairs of neurons, when this force is varied for each pair of neurons based on their mass and the euclidean distance between them. The mass of each neuron represents the number of times which this neuron selected as winner neuron. Let $\mu_i(t)$ be the mass of the i_{th} neuron after step t where $\forall_{i \in \Omega}(\mu_i(0) = 1)$ at the initial state. Then, for each time t , the mass of the winner neuron $\nu(t)$ is increased by one unit. Therefore, the winner will be the neuron generating more gravitational attraction force to the selected point instead of the closest neuron. For this purpose, we substitute the euclidean distance with gravity force in both Equations 5.3 and 5.6 respectively:

$$\nu(t) = \operatorname{argmax}_{k \in \Omega} \left[\frac{\mu_k(t)}{\|x(t) - v_k(t)\|^2} \right] \quad (5.16)$$

$$\eta(\nu(t), k, t) = \exp \left[\frac{\mathcal{F}(\nu(t), k, t)}{2\sigma(t)^{-2}} \right] \quad (5.17)$$

Where the mass of the presented input $x(t)$ is equal to one and $\mathcal{F}(\nu, k, t)$ is the gravity force between node ν and k :

$$\mathcal{F}(\nu, k, t) = \frac{\mu_\nu(t) \cdot \mu_k(t)}{\|r_\nu - r_k\|^2} \quad (5.18)$$

Fig.5.3b shows an example where the dense cluster $C8$ (orange), with big size and small standard deviation, curved the FSC dramatically while the sparse one $C1$ (yellow) only slightly bent it.

We empirically figure out the appropriate value for the time constants $\lambda_\alpha = 1000/\alpha_0$ and $\lambda_\sigma = 1000 \cdot \sigma_0$ by conducting several experiments. These values lead the GSOM to generate the smooth FSC without crumpling and folding that is necessary for the transformation step of our approach.

5. New Method to Homogenize the Density

There are two principal consequences of this gravitation-based function: (1) the updated weight of neurons depend on both their distances and masses, which guarantees the smoothness of the final FSC, and (2) the GSOM converges faster to stable model when the massive neurons stabilize earlier than light mass neuroses which arrange around them later on, in the same way as the stars in a galaxy.

5.5.1.4 Early Stopping Condition

In our approach, we use MDCU to learn convergence mechanism per each iteration rather than calculate after the training process like the standard SOM. We compute the $MDCU(t)$ per each iteration t , as follows:

$$MDCU(t) = \frac{1}{n} \sum_{i=1}^n \min_{k \in \Omega} \|x_i - v_k(t)\| \quad (5.19)$$

It considers the stopping condition that defines the proper GSOM's status as no training samples reduces the overall MDCU. However, a more elaborate trigger is required in practice since the training of the GSOM is stochastic that can be noisy. Therefore, GSOM monitors the overall MDCU for a given number of consecutive epochs in order to try to avoid falling into a local minima. Using this condition, the training process is automatically stopped as soon as it successively sees no reduction in MDCU metric over a given number of epochs \mathcal{E} where $\mathcal{E} = 10$ works well based on our conducted experiments. We compute early stop function $ES(t)$ per each iteration t , as follows:

$$ES(t) = |\{e \in E | MDCU(t - e) - MDCU((t - 1) - e) \leq 0\}| \quad (5.20)$$

where $t > \mathcal{E}$ and $E = [0, \dots, \mathcal{E}]$. The training is stopped as soon as the $ES(t)$ is equal to \mathcal{E} .

Note that the FSF modeling divides up the feature space into regular rectilinear cells to capture the density structure of data, see Fig.5.4a. Although, the computed FSC's cells are not longer rectilinear but are irregular quadrilaterals, see Fig.5.4b.

As a result, by applying the GSOM on X , we obtain an m-dimensional irregular grid \mathbb{T} in size ς and $V = [v_1, v_2, \dots, v_\varsigma]$, $v_i \in R^m$, which accurately represents the density structure of the clusters by preserving the topological structure of data. We summarize the FSC modeling algorithm in Algorithm 2.

5.5.2 Curvature Map

In order to apply traditional density-based clustering algorithms which are designed for euclidean spaces, we need to project the data from a non-Euclidean curved space FSC

5. New Method to Homogenize the Density

Algorithm 2 : Feature Space Curvature modeling

Input: n data points with m features;
 σ_0 : The initial neighborhood size;
 α_0 : The initial learning rate;
 ξ : The number of grid lines;
 \mathcal{E} : The validation number of epochs;
Output: The grid \mathbb{T} and the neurons $V = [v_1, v_2, \dots, v_\zeta]$;

- 1: Initiate RRG map \mathbb{T} of size ξ . (Eq.5.13)
- 2: **repeat**
- 2.1: Select $x_i \in X$ randomly
- 2.2: Find the winner neuron $\nu(t)$ (Eq.5.16)
- 2.3: Update weight of the winner and its neighbors (Eq.5.17)
- 2.4: $\mu_{\nu(t)} = \mu_{\nu(t)} + 1$
- 2.5: Decrease the learning rate $\alpha(t)$ (Eq.5.5)
- 2.6: Decrease the neighborhood size $\sigma(t)$ (Eq.5.7)
- 3: **until** $ES(t) < \mathcal{E}$ (Eq.5.20)
- 4: **return** $\mathbb{T}, V = [v_1, v_2, \dots, v_\zeta]$

to the new euclidean flat space. Our transformation mechanism reduces the density of high-density regions intensively while it slightly reduces the density of low-density ones.

As euclidean space, it is described by orthogonal dimensions and its discretized version can be represented by a rectangular grid FSF. The resulting FSC is described by a deformed grid where the nodes in the original coordinate space now form irregular quadrilaterals. Our data transformation method "relinearizes" the FSC considering its original grid structure and performing a multilinear projection of each original data point to the new euclidean grid space based on its position in its enclosing irregular quadrilateral of the FSC.

Let $U' = [u'_1, \dots, u'_\varphi]$ be the initial position of grid \mathbb{T} cells in m -dimensional space, where $\varphi = (\xi - 1)^m$, $u'_i \subset V$ and $|u'_i| = 2^m$; and $L = [\ell_1, \dots, \ell_m]$ stand for the side sizes of each regular cells where $\ell_i = |I_i| / (\xi - 1)$. In the same way, $U = [u_1, \dots, u_\varphi]$ denotes the cells position in FSC when $u_i \rightarrow u'_i$.

We begin by computing $\mathcal{B} = [\beta_1, \dots, \beta_\varphi]$ a set of parametric transformation function between FSC and FSF. For each cell $u_j \in U$, we compute individual transformation function β_j by giving the displacement vectors $\vec{D} = [\vec{d}_1, \dots, \vec{d}_{2^m}]$ of the cell corners, see section 5.4.2.

Then, for each $x_i \in X$, we identify the cell $u_\phi \in U$ which x_i is located inside it. In order to do that, we firstly find the Best Unit Match (BMU) $\nu(x_i)$ by applying Equation 5.3.

According to the \mathbb{T} topology, the neighborhood $\mathcal{N}(\nu(x_i))$ in R^m is split into 2^m cells. If the direct topological neighbors of the $\nu(x_i)$ expresses by $\Theta = [\theta_1, \dots, \theta_{2^m}]$

5. New Method to Homogenize the Density

where $\theta \subset \Omega$. Then we can specify the u_ϕ with $\nu(x_i)$ and m lattice indicator points in the neighborhood. Therefore, to single out the cell $u_\phi \in U$ which x_i is located inside, we find the most similar lattice point $l_0 \in \Theta$ to x_i :

$$l_0 = \operatorname{argmax}_{k \in \Theta} (\vec{\mathcal{P}}(x_i) \cdot \vec{\mathcal{P}}(\theta_k)) \quad (5.21)$$

where $\vec{\mathcal{P}}(\theta_k) = \vec{v}_{\theta_k} - \vec{v}(x_i)$ is the position vector of the lattice point θ_k , similarly $\vec{\mathcal{P}}(x_i) = \vec{x}_i - \vec{v}(x_i)$ is the position vector of x_i . Then, to distinguish the source cell $u_\phi \in U$, we complete the lattice indicator set as follows:

$$L_{x_i} = \{k \in \Theta : \vec{\mathcal{P}}(l_0) \cdot \vec{\mathcal{P}}(\theta_k) \leq \vec{\mathcal{P}}(x_i) \cdot \vec{\mathcal{P}}(\theta_k)\} \quad (5.22)$$

After we get cell lattice indicator set L_{x_i} , we could find the cell u_ϕ easily since we keep the grid topology \mathbb{T} . Finally, for each $x_i \in X$ we apply the appropriate multilinear transformation $\beta_\phi(x_i)$ to map it into the regular FSF.

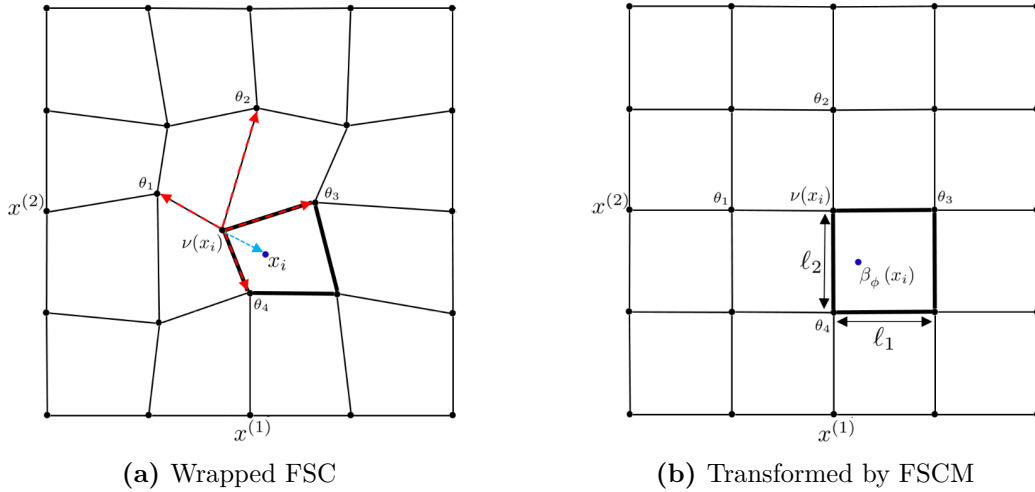


Figure 5.4: Data point transformation between a bent FSC (a) and a regular FSF (b) based on the Multilinear Mapping in \mathbb{R}^2 .

Fig.5.4 shows a 2D grid example where its cells are represented by 2^2 corners. Therefore, to map the wrapped FSC (Fig.5.4a) onto the regular FSF (Fig.5.4b), we compute the transformation function \mathcal{B}_j for each cell u'_j , where $\vec{d}_{j1}, \vec{d}_{j2}, \vec{d}_{j3}, \vec{d}_{j4}$ are the displacement vectors of cell corners. The $\nu(x_i)$ is the BMU. The red and blue arrows represent the position vectors of lattices and position vector of x_i respectively. The computed indicator set is $L_{x_i} = \{\theta_4, \theta_3\}$ where $l_0 = \theta_4$. As a result, the cell u'_ϕ which x_i is located inside (the thick block) is depicted with this lattice set. Finally, $\mathcal{B}_\phi(x_i)$ shows the transformer of x_i after the applying mapping function $\mathcal{B}_\phi : u'_\phi \rightarrow u_\phi$. We summarize the Curvature Map algorithm in Algorithm 3.

5. New Method to Homogenize the Density

Algorithm 3 : Curvature Map

Input: n data points with m features;
The grid \mathbb{T} and the neurons $V = [v_1, v_2, \dots, v_\zeta]$;
Output: Homogenized data $X' = (x'_1, \dots, x'_n)$, $x'_i \in R^m$;

- 1: Compute the regular cell side size $L = [\ell_1, \dots, \ell_m]$
- 2: Compute the initial position of cells $U' = [u'_1, \dots, u'_\varphi]$
- 3: Compute the final position of cells $U = [u_1, \dots, u_\varphi]$
- 4: **for** $\phi = 1$ to φ **do**
- 5: Compute the displacement vectors $\vec{D} = [\vec{d}_1, \dots, \vec{d}_{2^m}]$
- 6: Compute the map function $\mathcal{B}_\phi : u_\phi \rightarrow u'_\phi$ (Sec. IV-A)
- 7: **end for**
- 8: **for** $i = 1$ to n **do**
- 9: Find the BMU $\nu(x_i) = \operatorname{argmin}_{k \in \Omega} \|x_i - v_k\|$
- 10: Compute the indicator set L_{x_i} of source cell u_ϕ (Eq.5.22)
- 11: Find the source cell u_ϕ
- 12: Compute the transformation $x'_i = \beta_\phi(x_i)$
- 13: **end for**
- 14: **return** $X' = (x'_1, \dots, x'_n)$

5.6 Application of FSCM in the Real Data

In this section, we have carried out several experiments to show the efficiency and effectiveness of our proposed FSCM method to overcome the weaknesses of existing density-based clustering algorithms in diagnosing all clusters with varying densities.

5.6.1 Datasets

We used 10 real-world datasets from the UCI Repository¹ and a synthetic datasets Syn10, to validate the capability of our method in homogenizing multi-density dataset. Table.5.1 outlines the basic properties of the datasets.

Syn10 is syntactic "hard distributed" 2-dimensional data that is generated by sampling a mixture of 10 Gaussian distributions $N(\mu, \sigma)$. Therefore, this data set is labeled and can be used to measure the quality of the different clustering approaches. The statistics $(size/\mu(x^{(1)}, y^{(2)})/\sigma)$ of each cluster was shown in Fig.5.3a. As shown in the density plot Fig.5.5a, the clusters do not satisfy the clause stated in the Equation 5.2. As a result, DBSCAN fails to precisely stratify the clusters. For example, the 3 clusters at the top-right and other 3's at bottom of the density plot represent high density areas which DBSCAN is unable to separate them by applying an appropriate global *eps* value, see Fig.5.5b.

¹<https://archive.ics.uci.edu>

5. New Method to Homogenize the Density

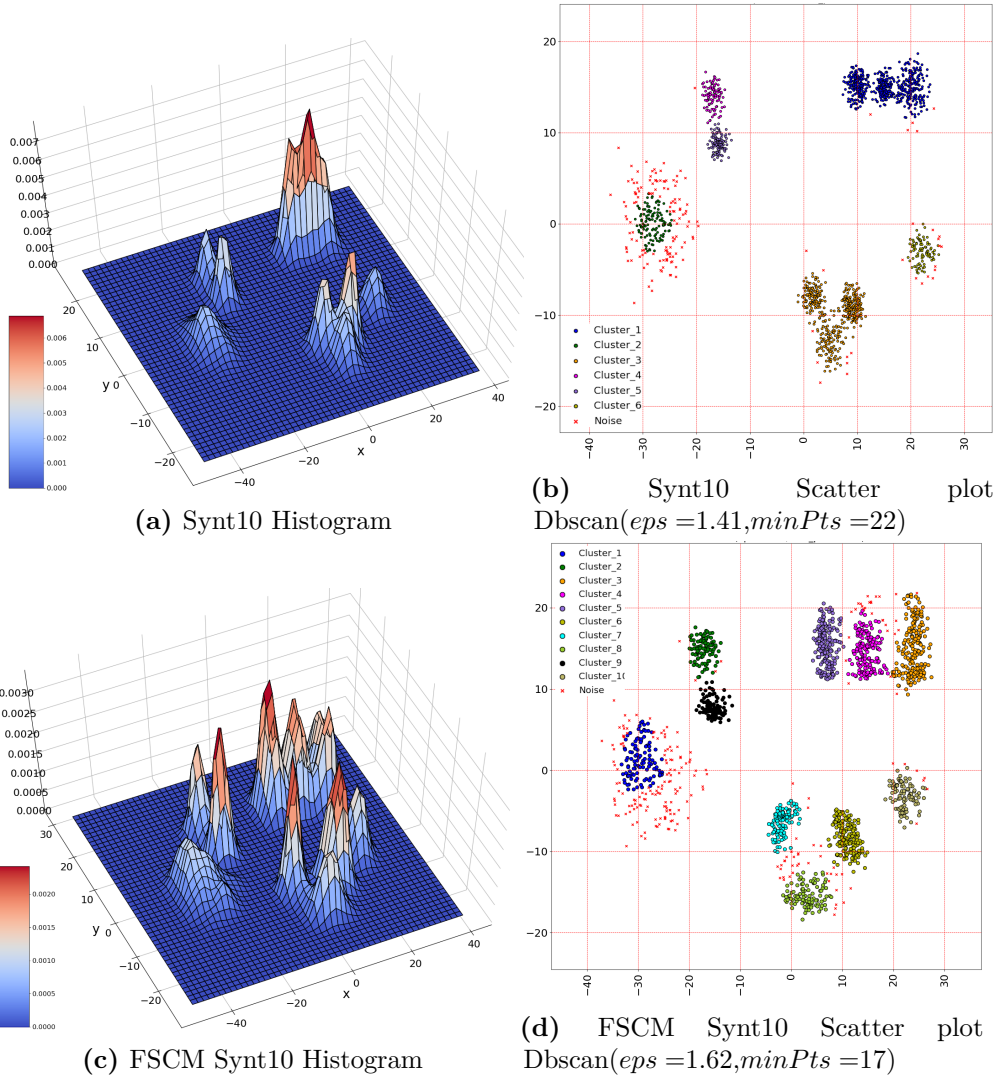


Figure 5.5: Comparison of 3D original histogram (a) of dataset Syn10, previously shown in Fig.5.3a, and the homogenized density (c) by our FSCM method, subsequently their DBSCAN clustering results (b) and (d). Without applying FSCM, DBSCAN ends up merging the three blobs in the top right into a single one in order to be able to identify some cluster point in the sparse blob on the left. FSCM as a preprocessing step to DBSCAN allows to better identify the overall structure of the data.

5.6.2 Evaluation Metric

We use the Overall F-measure denoted by “FScore” which is commonly used in the literature to compare the quality of clustering result. $FScore$ is computed by the harmonic mean of precision score and recall score, and the overall $FScore$ is the unweighted average over all clusters:

$$FScore = \frac{1}{q} \sum_{i=1}^q \frac{2P_i R_i}{P_i + R_i} \quad (5.23)$$

Where P_i and R_i are the precision score and the recall score of the cluster c_i respectively. The higher the value of $FScore$ is, the better the clustering performance is.

Table 5.1: Datasets properties

Dataset	Points (n)	Features (m)	Classes (q)
Segment	2310	19	7
Wine	178	7	3
Wifi	2000	7	4
Iris	150	4	3
Breast	569	30	2
ForestType	523	27	4
Diabetes	768	8	2
Ecoli	336	7	8
Pendig	10992	16	10
ILPD	579	9	2
Syn10	1530	2	10

5.6.3 Experiment Setup

Our method, denoted here by FSCM, is compared with: both DScale and ReScale, which consists of first using these scaling algorithms for data preprocessing, and then implementing three state-of-the-art density-based clustering algorithms (DBSCAN, OPTICS and DP) to partition the data.

Before the experiments began, we normalized each dataset by scaling each feature to $[0,1]$ range by min-max normalization. To be able to visualize both our methodology and the empirical result in 2D, we reduced the data dimensionality by applying Principal Component Analysis (PCA). Then, we conducted the whole set of experiments by taking into account the first two Principal Components. We apply the Exhaustive Grid Search method to search the hyper-parameter space for the best validation *FScore*. Table.5.2 specifies the parameters and their search space for each algorithm. The search ranges of both ψ and η are as used by [216] where ψ is the number of intervals to estimate and control the precision of \hat{f}_η . We implemented entire set of algorithms used in our experiments in python.

Table 5.2: Parameters and their search ranges.

Algorithm	Parameters & Search Range
DBSCAN	$\epsilon \in [0, 2], \text{minPts} \in \{1, 2, \dots, 25\}$
OPTICS	$\epsilon \in [0, 2], \text{minPts} \in \{1, 2, \dots, 25\}$
DP	$\epsilon \in [0, 2], k \in \{1, 2, \dots, 20\}$
ReScale	$\psi = 100, \eta \in \{0.1, 0.2, \dots, 0.5\}$
DScale	$\eta \in \{0.1, 0.2, \dots, 0.5\}$
FSCM	$\xi \in \{3, \dots, 10\}, \alpha_0 \in \{0.005, 0.006, \dots, 0.015\},$ $\sigma_0 \in \{1, 1.1, \dots, 2.0\}$

Table 5.3: The best $FScore$ for DBSCAN, OPTICS, DP, and their ReScale, DScale and FSCM versions. For each clustering algorithm, the best performer in each dataset is boldfaced. Orig, PCA, ReS, and DS represent the Original algorithm, Principal Component Analysis, ReScale, and DScale respectively.

Dataset	DBSCAN					OPTIC					DP				
	Orig	PCA	ReS	DS	FSCM	Orig	PCA	ReS	DS	FSCM	Orig	PCA	ReS	DS	FSCM
Segment	0.59	0.62	0.62	0.61	0.70	0.69	0.69	0.67	0.70	0.74	0.78	0.78	0.77	0.80	0.85
Wine	0.64	0.65	0.86	0.80	0.91	0.76	0.78	0.84	0.88	0.91	0.93	0.98	0.95	0.96	0.97
Wifi	0.74	0.76	0.87	0.86	0.90	0.79	0.76	0.88	0.85	0.93	0.90	0.91	0.92	0.92	0.95
Iris	0.85	0.89	0.90	0.93	0.96	0.85	0.85	0.84	0.88	0.94	0.97	0.97	0.97	0.97	0.97
Breast	0.82	0.82	0.95	0.96	0.94	0.84	0.79	0.96	0.95	0.95	0.97	0.79	0.97	0.97	0.97
ForestType	0.27	0.32	0.51	0.48	0.49	0.29	0.51	0.64	0.52	0.61	0.69	0.75	0.83	0.70	0.80
Pima	0.43	0.46	0.48	0.64	0.72	0.65	0.65	0.65	0.66	0.70	0.62	0.64	0.66	0.67	0.71
Ecoli	0.37	0.42	0.40	0.54	0.53	0.44	0.50	0.57	0.50	0.51	0.48	0.53	0.55	0.63	0.61
Pendig	0.70	0.73	0.78	0.74	0.78	0.74	0.78	0.78	0.78	0.81	0.79	0.79	0.82	0.82	0.84
ILPD	0.41	0.42	0.42	0.56	0.57	0.47	0.47	0.47	0.57	0.54	0.60	0.60	0.63	0.62	0.63
Syn10	0.66	0.61	0.58	0.72	0.89	0.67	0.65	0.65	0.76	0.91	0.69	0.70	0.60	0.78	0.92
Average	0.59	0.60	0.67	0.71	0.77	0.65	0.68	0.72	0.73	0.78	0.77	0.78	0.79	0.81	0.85
Winner %	0.0	0.0	9.1	18.1	72.8	0.0	0.0	27.3	9.1	63.6	0.0	9.1	9.1	9.1	72.7

5.6.4 Clustering Results

The results obtained in our experiments are shown in Table 5.3. The highest obtained *FScore* values for each dataset-algorithm are highlighted in bold. The second last row reports the average *FScore*. As we can see, our proposed FSCM method persistently surpasses both competitors on all the three clustering algorithms. The highest magnitude of improvement depicted for DBSCAN from 0.59 to 0.77. However, for DP and OPTICS, the performance gap shrinks slightly since they are advanced version of DBSCAN which are using multiple density thresholds to efficiently detect cluster centers. Still FSCM increases the clustering performance of DP and OPTICS significantly more than both ReScale and DScale.

The last row of Table 5.3 shows the proportional performance winner. It is computed by the number of times a preprocessing method wins the performance divided by total number of datasets for each clustering algorithm. By looking at the first five columns, we see that FSCM-DBSCAN outperforms the other two methods in a large majority, 8 out of 11, of the datasets (in many cases by a large margin); while FSCM-OPTICS and FSCM-DP are the performance winner on 7 and 9 out of 11 datasets, respectively. In the only three datasets where FSCM does not perform best (Breast, ForestType and Ecoli), its *FScore* values are very close to the “winner”.

In case of Syn10, our FSCM remarkably improves the clustering performance of all existing algorithms, probably because the data is generated from a mixture of Gaussian sources. For example, in Fig.5.5c we present the histogram of the homogenized Syn10 by applying our FSCM. The mapping yields that both three sets of clusters in the top-right and bottom of original space (corresponding to the two high density reigns, see Fig.5.5a) are expanded and segregated in the new space with valleys appearing between them. As a result, the densities at the peaks of different clusters are closer after this mapping. Thereafter, we could efficiently stratify 10 distinct clusters which correspond to the actual generative model for the data by using DBSCAN, see Fig.5.5d. However, ReScale fails to increase clustering performances on this dataset since it is just using one-dimensional scaling. Furthermore, the DScale moderately increased the *FScore* values of DBSCAN, OPTICS and DP about 0.06, 0.09 and 0.09 unit respectively, since it still depends on a rescaled distance. On the other hand, our FSCM method significantly improved *FScore* values of DBSCAN, OPTICS and DP from 0.66, 0.67 and 0.69 to 0.89, 0.92 and 0.92, respectively.

In Fig.5.6 and Fig.5.7, we compare the DBSCAN clustering quality on two original WiFi and Breast datasets and their transformation due to FSCM. They show that both the transformed datasets are stratified more efficiently than the original data by DBSCAN. For the WiFi dataset, see Fig.5.6, plot(a) shows the original data with

5. New Method to Homogenize the Density

original labeling, including four distinct classes, and the embedded FSC computed by FSCM. As we can see that the *Class_2* (red) is dense while the *Class_1* (pink) is sparse. As shown in plot (b), DBSCAN just identified three clusters due to the fact that we observed varied densities in data. Consequently, classes 2 and 4 are jointly identified as a single cluster, see the blue cluster in Fig.5.6b. In contrast, *Class_2* becomes sparser using FSCM and we generally observe that the projected data has approximately uniform densities in the populated areas, as shown in plot (c). As a result, DBSCAN identified four distinct clusters by applying single density threshold, as shown in plot (d). Further, our FSCM method significantly improved *FScore* values of DBSCAN from 0.76 to 0.9, see Table.5.3.

For the Breast dataset, see Fig.5.7, plot(a) shows the original data with original labeling, including two distinct classes, and the embedded FSC computed by FSCM. As we can see that the *Class_1* (dark blue) is significantly dense while the *Class_2* (light blue) is sparse. As shown in plot (b), DBSCAN just identified the dense cluster. Consequently, the majority of the data points belonged to the sparse class are designated as noise, the red \times . In contrast, *Class_1* becomes sparser using FSCM and we generally observe that the projected data has approximately uniform distribution, as shown in plot (c). As a result, DBSCAN identified two distinct clusters by applying single density threshold, as shown in plot (d). As illustrated in Table.5.3, our FSCM method significantly improved *FScore* values of DBSCAN from 0.82 to 0.96.

Table 5.4: Complexity of ReScale, DScale and FSCM.

Algorithm	Time complexity	Memory complexity
ReScale	$\mathcal{O}(mn\psi)$	$\mathcal{O}(mn + m\psi)$
DScale	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn + n^2)$
FSCM	$\mathcal{O}(t_f(m\varsigma + n\varsigma^2))$	$\mathcal{O}(m\varsigma + n\varsigma^2)$

5.6.5 Complexity Analysis

The computational complexity of ReScale, DScale and FSCM are shown in Table 5.4. The result shows DScale has higher computational complexity than ReScale since it computes and frequently updates a $n \times n$ distance matrix. While the computational complexity of FSCM is roughly corresponding to the performance of Feature Space Curvature modeling phase which is similar to standard SOM algorithm. Considering a map of ς neurons and the input data $X \in R^{n \times m}$, then each learning epoch $t \in [1, \dots, t_f]$ of the GSOM algorithm costs $\mathcal{O}(m\varsigma + n\varsigma^2)$ elementary operations. Therefore, its overall theoretical complexity is $\mathcal{O}(t_f(m\varsigma + n\varsigma^2))$. The size of map ς is the only parameter which could be quite big for the high-dimensional data.

5. New Method to Homogenize the Density

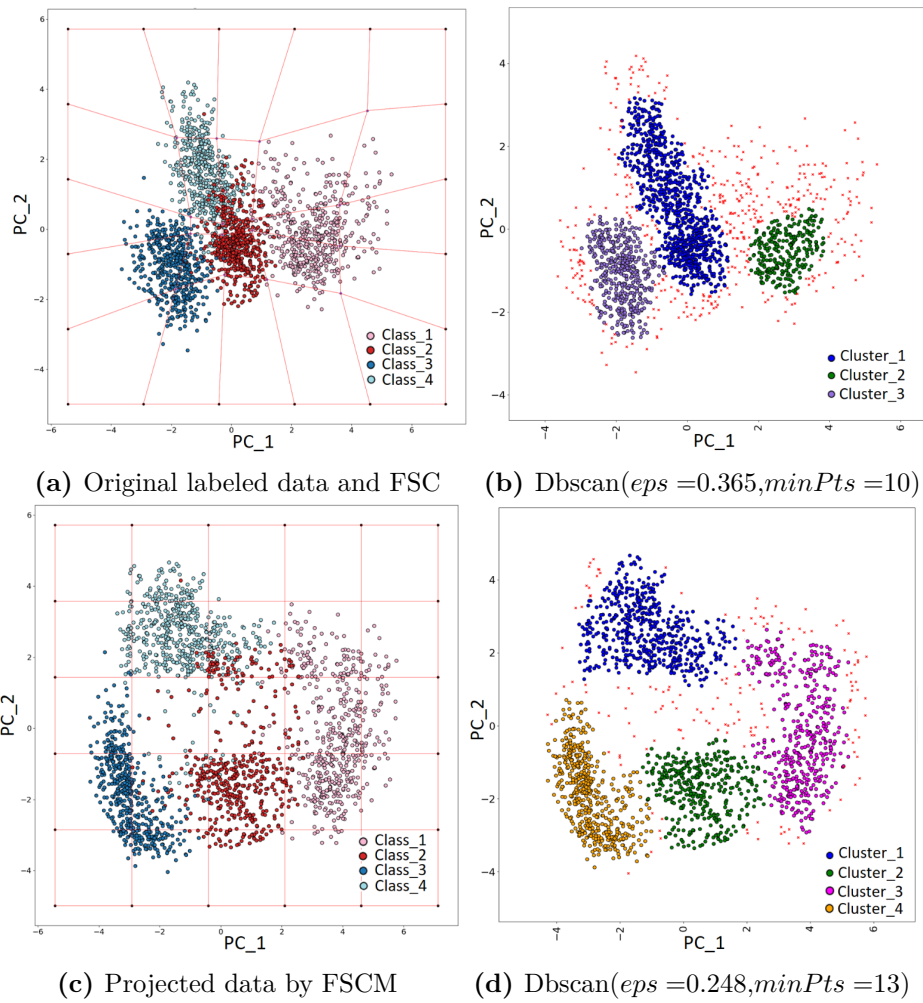


Figure 5.6: Application of FSCM-DBSCAN on the Wifi dataset and the original datasets in \mathbb{R}^2 .

Furthermore, the computational complexity of the majority of existing density-based clustering algorithms is $\mathcal{O}(n^2)$ [217], thus our method doesn't notably affect their final complexities.

5. New Method to Homogenize the Density

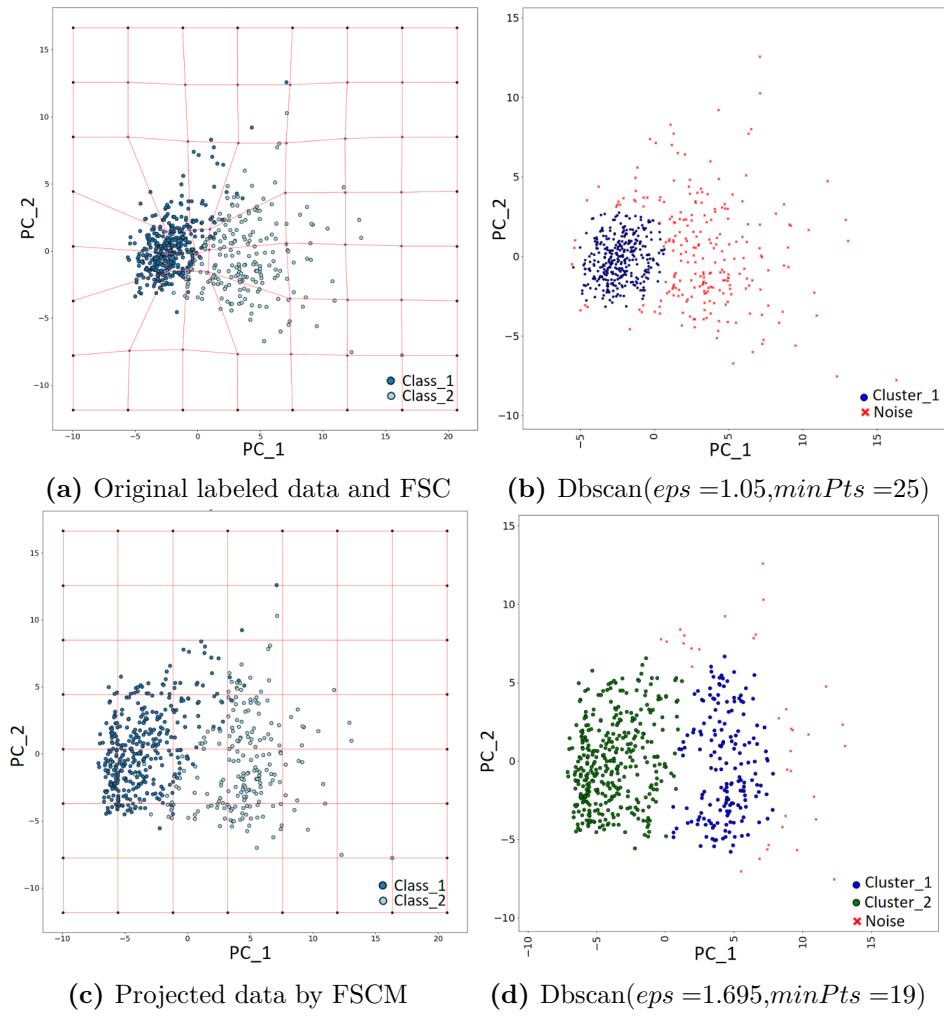


Figure 5.7: Application of FSCM-DBSCAN on the Breast dataset and the original datasets in \mathbb{R}^2 .

6

New Method for Extracting Insights from the Shape of Cluster

Contents

6.1	Introduction	101
6.2	Related work	104
6.3	Background and Notation	105
6.4	Organization Component Analysis	107
6.5	Application of OCA in the Real Data	110
6.5.1	Parameter Selection	111
6.5.2	Evaluation Quality of the Map	111
6.5.3	Identifying Spatial Patterns of Wilderness Sub-area.	111
6.5.4	Diagnosing Performance Bottleneck in HPC Applications.	115

In this chapter, we present a novel topological-based method to extract insight from the clustering result. This method enables us to interpret the result of clustering mathematically by the explanation of internal variability of each cluster.

6.1 Introduction

Cluster analysis is used as a matter of course throughout the experimental sciences to extract scientific information from data[220]. But it turns out that many well-performed segmentation results cannot be turned into profound insights easily. Because the process of making clusters is a generic mathematically oriented process but lacks the intuition and domain knowledge that is often required to interpret and drill down into the algorithmic results. However, cluster "Geometric Features" can still

6. New Method for Extracting Insights from the Shape of Cluster

be disclosed from metrics on the dataset, which could represent mathematically the fine-grain structure of complex data.

Geometric features of data can reveal clusters of diverse shapes, sizes and densities as demonstrated in Fig.6.1. Clusters can be spherical (a), elongated (linear) (b), loop (c), tendril (d), and heterogeneous (e) [221]. We are interested in studying such features of data since we assume insight into the shape of scientifically relevant data, it has a good chance of giving insight into the science itself. Experience has shown that this assumption is a reasonable one. For example, the study of loops and their higher-dimensional analogues has recently offered insight into questions in biophysics[222] and natural-scene statistics[223]; and, the study of tendrils has recently offered insight into oncology[224]. However, aforementioned figures say we should not select a final set of model types and then we build individual model, but we should make modeling mechanism that can study all arbitrary shapes and computed easily. Therefore, the basic goal of this chapter is to introduce a generalized method for studying geometric features of clustered data.

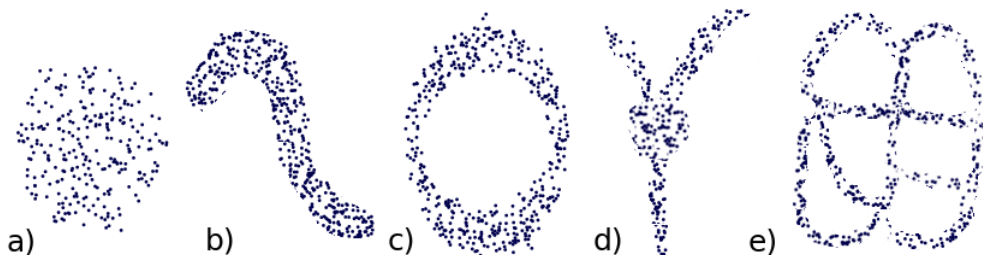


Figure 6.1: Clusters of diverse shapes in \mathbb{R}^2

In our methodology, we propose a novel clustering result interpretation method by combining techniques from algebraic topology and statistical learning to give a quantitative basis for the study of the geometric features (shape) of a data cluster. It learns to interpret a data cluster with correct topology preservation map as modeling mechanism that doesn't make assumptions about the form of the mapping function. As a result, we can generally apply it to study the geometric feature of arbitrary cluster-shape. Moreover, this topology preservation map describes the idea of closeness in terms of relationships between sets rather than euclidean distance in the feature multidimensional space. The key objective is that once we represent high dimensional data by mapping functions on small space, we can efficiently give direct insight into the data.

Another important factor is the objective of clustering algorithm. Since various discriminative definitions of a cluster-shape can be formulated, depending on it.

6. *New Method for Extracting Insights from the Shape of Cluster*

For example, Density-based clustering such as DBSCAN[225], ISB-DBSCAN[226] or RNNDBSCAN[227] is famous for its capability of finding out arbitrary shape clusters from datasets. These approaches regard clusters as regions in the data space in which the data points are dense, and separated by regions of low data point density (noise). Since these methods apply a local cluster criterion to detect regions that may have an arbitrary shape and the data points inside a region may be arbitrarily distributed[228]. For such shape, it is very hectic to determine their innate structure or quantify them by applying correlation oriented techniques since the assumption of linear relationship between all variables behind these techniques do not often hold true for this arbitrary distributed data. Throughout this chapter, we are interested to extract insights from arbitrary cluster-shape by not making assumptions about linearity. To do this, we propose a novel Directional Sequence Similarity method to locally measure the non-linear relationships between features.

Furthermore, in practice, not all features are important and relevant to the overall clustering task, many of them are often correlated and redundant, which may result in adverse effects such as low efficiency and poor performance, and also, dramatically increasing computational cost. Feature selection is one effective mean to identify relevant features for dimension reduction [229]. Once a reduced (active) feature subset is chosen, conventional clustering algorithm can then be applied by using the active features. Consequently, unselected (illustrative) features remain untouched that can be applicable to interpret the local prototype of cluster. In our approach, we consider active features to model the overall clustering space and also illustrative features to interpret the local establishment of each cluster with maximum comprehensiveness.

In this chapter, we introduce a new method for the purpose of interpreting the clustering result through cluster-shape study which does not have assumption about data either distribution or shape. Our modeling mechanism organizes an augmented structure of a data cluster to represent its arbitrary shape by applying topology to develop tools for studying qualitative features of a cluster. This cluster-organization contains information which is equivalent to the topology-preserving map of finer-grained dense areas and their interconnection inside a cluster. We show how to efficiently and automatically interpret the topology-preserving map to extract not only the innate cluster structure, but also the causal factors associated with its formation by computing the non-linear local relationships between features. For that, we introduce a new Directional Sequence Similarity method to locally quantify it. Furthermore, we use illustrative features to obtain very detailed structure identification and a great detail on non-homogeneous local geometrical space within the identified clusters by generic clustering algorithms with using the active features.

6. *New Method for Extracting Insights from the Shape of Cluster*

The rest of this chapter is organized as follows. The intuition motivating the cluster-shape interpretation is presented along with an overview of related work and methods in section 6.2. In section 6.3, the basic notions of cluster-shape interpreting and background techniques are defined. In section 6.4, our novel algorithm Organization Component Analysis (OCA) to decipher a cluster-shape with respect to its self-organizing map structure is presented. The experimental result of our OCA method for the purpose of cluster-shape studying is illustrated in section 6.5.

6.2 Related work

There are various ways to explore the cluster-shape; through more rigorous analysis or by visualization and human interaction or external knowledge-based supervision or Topological Data Analysis.

Numerous statistical analytics techniques exist to study the multivariate data and the shape of cluster. Gaussian mixture model[230] is a probabilistic model for representing normally distributed subpopulations within an overall population. It can describe the shape of cluster as a sequence of overlapped subpopulations which have Gaussian distribution. Furthermore, in [231, 232], they discuss two multivariate analysis procedures: PCA and exploratory factor analysis, which can extract the latent structure of data. These techniques can identify a small set of synthetic variables, called eigenvectors or factors, that explain most of the total variation presented in the original variables and the shape of cluster. However, there are several requirements for a dataset such as: normality, homoscedasticity, linearity, sampling adequacy and no significant outliers. Taking advantage of the self-organizing map technique, we developed an assumption-free and efficient cluster-shape analysis method.

The other approach is visual analysis that is usually a very intuitive and manual way to explore the underlying structure of the data, possibly incorporating human feedback into the process. HD-Eye method [233] explores different subspaces of the data in order to determine clusters in different feature-specific views of the data. IPCLUS [234] generates feature-specific views in which the data is well polarized. A polarized data is a 2D subset of features in which the data clearly separates out into clusters. Then a kernel-density estimator determines the views in which the data is well polarized. Finally, the shape of cluster is defined by exploring different views of the data. These methods are intuitive which make it difficult to separate the definition of cluster from the perception of an end-user and even to automatize them. Our approach mathematically combines the strengths of statistical and topological methods to eliminate the need for expert human visual analysis.

6. *New Method for Extracting Insights from the Shape of Cluster*

Supervision also can play an effective role, because it takes the specific goal and subjectivity of the analyst into consideration, which leads our insight of cluster-shape. In [235], they propose an interactive approach to constrain clustering in which the user can iteratively provide constraints as feedback to refine the clusters towards the desired concept. The results indicate that significant profounder insight can be made with only a few well-chosen constraints. Also, in [236], they describe an expectation maximization (EM) algorithm, penalized probabilistic clustering, which interprets pairwise constraints as prior probabilities that two items should, or should not, be assigned to the same cluster. This formulation permits both hard and soft constraints allowing users to specify background knowledge even when it is uncertain or noisy. Normally, the supervision should be embedded inside the clustering algorithm, which lead to poor generalization and automation. Our proposed Organization Component Analysis method is not dependent on a particular clustering algorithm, therefore any clustering result can be analyzed by it efficiently.

Topological Data Analysis (TDA) is a recent and fast-growing field providing a set of new topological and geometric tools to study shape of possibly complex data [237]. In [238], they purpose the Mapper that is a mathematical tool to identify shape characteristics of datasets by applying topological method. The Mapper identifies local clusters within the data and then it studies the interaction between these small clusters by connecting them to form a graph whose shape captures aspects of the topology of the dataset. Mapper graphs associated to datasets preserve a wealth of information about the original shapes, but it is computationally expensive especially for massive datasets and its size grows rapidly with the number of data points, for that reason, Mapper takes transposed data matrix to build topological model. In our method, we apply the self-organizing map (SOM) to efficiently learn and build a topology-preserving mapping that projects multi-dimensional data onto a lower 2D space by preserving the neighboring relations of the data.

The intention of our work is to purpose a novel generalized technique that can study arbitrary shape of clusters by leveraging of self-organizing map and topology data analysis; thereby, that is an assumption-free and automated, and it does not dependent on any particular clustering algorithm. Especially, in our approach we apply topology-preserving mapping to optimize the computation cost and increase the efficiency.

6.3 Background and Notation

In this section, we discuss an overview of technique that we use leverage in our analytic approach. We firstly illustrate a brief mathematical notation, then we revisit the self-organizing map algorithm.

6. New Method for Extracting Insights from the Shape of Cluster

We use $D = (d_1, d_2, \dots, d_Q)$ to indicate a full dataset where $d_i \in R^M$. We suppose an active feature set M_a is selected and the rest of features M_l are illustrative ones, where $\forall M_a, M_l \subset M, M_a \cap M_l = \emptyset$ and $M_a \cup M_l = M$. Then, clustering algorithm (e.g. DBSCAN) is applied by using the active features to partition the Q observations into $h(\leq Q)$ clusters $\mathcal{CL} = \{cl_1, cl_2, \dots, cl_h\}$ with discretionary shape. In this chapter, we are interested in interpreting the shape of a particular cluster $X = (x_1, x_2, \dots, x_N)$ where $X \in \mathcal{CL}, x_i \in R^M$ and $N \leq Q$.

Self-organizing map

Kohonen's self-organizing map (SOM) is one of the most famous neural network models. Self-organizing map fundamentally is a pattern recognition technique in multivariate data, in which intra-pattern relations among the features are grasped without the attendance of a potentially biased or subjective external influence [239].

The SOM often arranged a set of neurons in a 2D rectangular or hexagonal grid \mathbb{T} in size n , to establish a discrete topological mapping of an input space, $X \in R^M$. Ω is the set of neuron indexes. The neurons are represented by set of weight vectors $V = \{v_1, v_2, \dots, v_n\}$, where v_i is the weight vector associated with neuron i and is a vector of the same dimension $-M-$ of the input, n is the total number of neurons, and let r_i be the location vector of neuron i on the grid. At the start of the learning, all the weights are initialized to small random numbers. Then the algorithm repeats next two steps until the map converges in order to preserve maximum topological properties of the data on the map [218].

First at each time-step t , presents an input $x(t)$ at random, and selects the winner neuron:

$$\nu(t) = \underset{k \in \Omega}{\operatorname{argmin}} \|x(t) - v_k(t)\| \quad (6.1)$$

Second, update the weights of the winner and its neighbors:

$$\Delta v_k(t) = \alpha(t)\eta(\nu, k, t)[x(t) - v_\nu(t)] \quad (6.2)$$

where η is the neighborhood function which Gaussian form is often used in practice – more specifically:

$$\eta(\nu, k, t) = \exp\left[-\frac{\|r_\nu - r_k\|^2}{2\sigma(t)^2}\right] \quad (6.3)$$

with σ representing the effective range of the neighborhood, and it is often decreasing with time.

The coefficients $\{\alpha(t), t \geq 0\}$, termed the 'adaptation gain', or 'learning rate', are scalar-valued that decrease monotonically, but satisfying:

$$0 < \alpha(t) < 1, \lim_{t \rightarrow \infty} \sum \alpha(t) \rightarrow \infty, \lim_{t \rightarrow \infty} \sum \alpha^2(t) < \infty \quad (6.4)$$

6. New Method for Extracting Insights from the Shape of Cluster

Furthermore, we apply "No Move" [239] to learn convergence mechanism in our approach. It considers stopping condition that defines no-improvement in SOM's status as no training samples changing their best match unit in a complete iteration of the training set. Using this condition, the training process is stopped as soon as it sees no-improvement.

As a result, the SOM can provide topologically preserved mapping [240] from input to output spaces, which includes grid \mathbb{T} and weight vectors V . For SOM training, the weight vector associated with each neuron moves to become the centroid of a local group of input vectors. The group i is represented by its centroid vector v_i and the local groups are connected via topological space \mathbb{T} . We use it for vector quantization by subdividing a subset of points into micro-clusters having points close to each other locally.

6.4 Organization Component Analysis

In this section, we will introduce our Organization Component Analysis (OCA) method which analyses the shape of a particular cluster X to extract the Organization Components $OC = (oc_1, oc_2, \dots, oc_C)$ where $oc_c \in R^{M_i}$ and $C \in \mathbb{Z} \cap [1, |M_a|]$.

First of all, train the SOM network to learn the topology-preserving map (\mathbb{T}, V) and structure of the input data X . The key observation is that neurons that are adjacent to each other in the topology should also move close to each other in the input space, therefore it is possible to explore a high-dimensional inputs space in the two dimensions of the network topology.

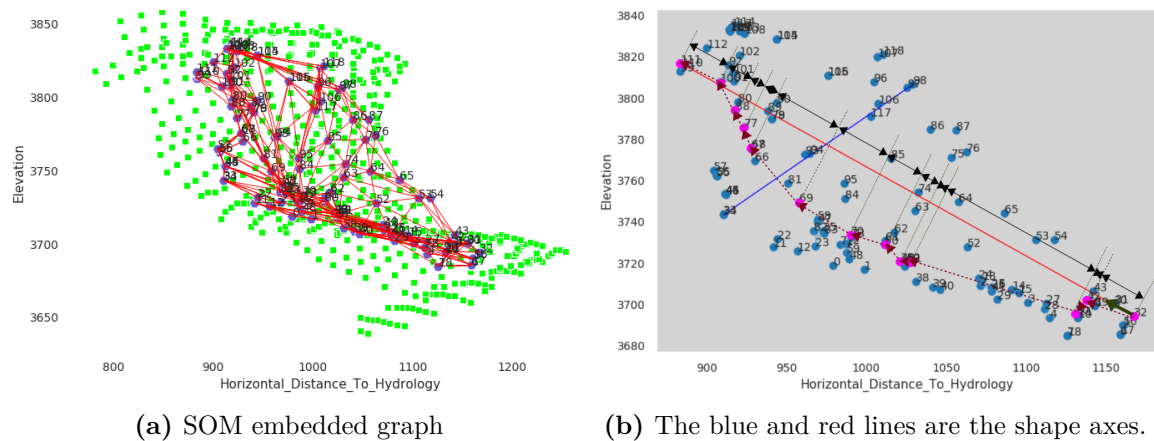


Figure 6.2: Application of OCA to spatial pattern indicator from the Covertypes dataset cluster C3, which is a part of our empirical study. The Horizontal_distance_to_hydrology and Elevation are selected active features. (a) shows the locations of the data points (green) and the weighted graph that is embedded in the SOM topology space. (b) The blue points are representing the SOM neurons, while the red and blue lines are major and minor axis, respectively. The rest of the items are described in the section IV in details.

6. New Method for Extracting Insights from the Shape of Cluster

In order to be able to compute pairwise relations between neurons, an undirected graph $G = (E, V, w)$ is embedded in the topological space \mathbb{T} . If G is represented in \mathbb{T} such that the vertices (V) of G are distinct elements in \mathbb{T} , and an edge (E) in G is a simple arc connecting its two ends such that E preserves the grid structure of \mathbb{T} , also $w : E \rightarrow \mathbb{R}$ is a function mapping edges to their values which is euclidean distance between two ends of each edge in the active feature subspace $\forall M_a \subset M$. See Fig.6.2.a, the labeled purple points and the red lines represent the nodes and edges respectively. Let $Va = [va_1, va_2, \dots, va_n]$, $va_i \in R^{M_a}$ be active feature subset, then the weight of edge between neuron i and j is;

$$w_{ij} = \| va_i - va_j \|, w_{ij} \geq 0 \quad (6.5)$$

Then we compute the axis I_c of the cluster. The axis is the $(va_o, va_q) \in R^{M_a}$ endpoints of the longest line that can be drawn through the cluster topological space \mathbb{T} . Let ep be a set of the endpoints where $ep = \emptyset$ during the computing of the first (major) organization component and it will be updated gradually. The endpoints $o \neq q$ of the I_c belongs to:

$$\operatorname{argmax}_{i,j \in \{\Omega \setminus ep\}} \left[\|va_i - va_j\| + \sum_{k \in ep} (\|va_i - va_k\| + \|va_j - va_k\|) \right] \quad (6.6)$$

And the axis and its unit vector are (e.g., the red solid line and green solid arrow in Fig.6.2.b),

$$I_c = \overrightarrow{va_o va_q}, \hat{I}_c = \frac{I_c}{\|I_c\|} \quad (6.7)$$

Then we update endpoints set:

$$ep = ep \cup \{o, q\} \quad (6.8)$$

Next, we describe the organization of the cluster in the direction of \hat{I} by capturing the continuing interaction between small local clusters (neurons) aligned with the I_c . To figure out this interaction we compute the geodesy path $P = (p_1, p_2, \dots, p_{n'})$ in the graph G (where $P \subseteq \Omega$, $p_1 = o$, and $p_{n'} = q$) by Dijkstra's algorithm, which is shown as pink nodes in Fig.6.2.b.

Technically, the relation between the neurons in path P can describe the major axis establishment in R^M . In order to characterize the relationship between the neurons in P , we compute a sequence of forward difference vectors $\Delta = [\delta_1, \delta_2, \dots, \delta_{n'-1}]$, $\delta_i \in R^M$ (Shown as red dotted arrows in Fig.6.2.b). Let $v_i \in R^M$ represent the weight vector of i_{th} neuron in P then

$$\delta_i \equiv |v_{i+1} - v_i| \odot v_i, \forall i \in (P \setminus \{p_{n'}\}) \quad (6.9)$$

6. New Method for Extracting Insights from the Shape of Cluster

If you imagine standing at i_{th} neuron in R^M , the vector δ_i tells you the changes rate in direction of $i + 1_{th}$ neuron.

We propose a Directional Sequence Similarity (DSS) method to compute the similarity between sequence of differences in active subspace R^{M_a} and in each illustrative feature on path P . Let A be sequence of difference in active subspace R^{M_a} aligned with I_c (The length of black intervals in Fig.6.2.b),

$$A = UL\Delta^T \quad (6.10)$$

Where L is diagonal matrix in size M with $l_{ii} = 1$ if $i \in M_a$ else $l_{ii} = 0$. To facilitate the computation, we define unit vector $U \in R^M$ with $u_i = \hat{I}_{ci}$ if $i \in M_a$ else $u_i = 0$.

Afterwards we compute sequential alignment between A and each $\delta'_i \in \Delta^T$, which are columns of Δ , to measure the relationship between directional sequence of changes in active subspace and each feature i belonged to illustrative subset M_l . Let $oc_c = [s_1, s_2, \dots, s_{M'}]$ be the associated organization component to the axis I_c , where $M' = |M_l|$ and s_i measures the sequence alignment between A and δ'_i :

$$s_i = \left| \frac{A \cdot \delta'_i}{\|A\| \times \|\delta'_i\|} \right| \quad (6.11)$$

Where $0 \leq s_i \leq 1$, a sequence is identical with A when $s_i = 1$ or $s_i = 0$ non-identical. And if a s_i is significantly similar to A , the i_{th} illustrative feature can be interpreted as an influencing feature on the organization component oc_c .

For any cluster, we can identify $|M_a|$ organization components by repeating these steps. The first organization component corresponds to the major discrete curve that passes through the multidimensional active feature space and represents the interrelationship among a chain of local micro-clusters in direction of main axis. The next organization components correspond to the same concept that its associated endpoints have been selected by maximization sum of their euclidean distance to previously selected endpoints ep in the active feature multidimensional space. These components can describe the fine-grain interconnection inside a cluster. And the most influencing feature in each organization component can illustrate the gravitational force among a chain of local micro-clusters. Through these organization components, we can often find fine-grain patterns in categorized data that traditional methodologies fail to find. For example in, Fig.6.2.b, the red dotted arrows show the first organization component composition that vividly represents the main repetitive patterns among the original data points (See green points in Fig.6.2.a).

We summarize the complete OCA algorithm for extracting insights from the shape of cluster in Algorithm (4).

6. New Method for Extracting Insights from the Shape of Cluster

Algorithm 4 : OCA for cluster shape interpretation

Require: N data points with M features;
 M_a : The active feature subset;
 M_l : The illustrative feature subset;
 n : Size of two-dimensional map;
 $C \in \mathbb{Z} \cap [1, |M_a|]$: Number of Components;

Ensure: $ep = \{\}$ endpoints
 $OC = \{\}$ Organization Compounds

- 1: Initiate SOM \mathbb{T} in size n and train it until converges as discussed in Section III.A. Let $V = [v_1, v_2, \dots, v_n]$, $v_i \in R^M$ contain neurons weight vectors.
 - 2: Embedding the $G = (E, V, w)$ in the \mathbb{T} and compute the w in R^{M_a} (Eq.6.5)
 - 3: **for** $c = 1$ to C **do**
 - 21.1: Select endpoints o, q of axis I_c (Eq.6.6)
 - 21.2: Compute axis I_c and unit vector \hat{I}_c (Eq.6.7)
 - 21.3: $ep = ep \cup \{o, q\}$. (Eq.6.8)
 - 21.4: Extract geodesy path $P = (p_1, p_2, \dots, p_{n'})$ between o and q in G by Dijkstra's algorithm.
 - 21.5: Compute sequence of forward difference vectors $\Delta = [\delta_1, \delta_2, \dots, \delta_{n'-1}]$. (Eq.6.9)
 - 21.6: Compute A sequence of changes in active subspace R^{M_a} aligned with I_c (Eq.6.10)
 - 21.7: Compute $oc_c = [s_1, s_2, \dots, s_{M'}]$ associated organization component to the axis I_c . Any s_i measures the sequence alignment between A and $\delta'_i \in \Delta^T$. (Eq.6.11)
 - 21.8: $OC \leftarrow OC + oc_c$
 - 4: **end for**
 - 5: **return** OC
-

6.5 Application of OCA in the Real Data

In this section, we apply OCA to two datasets from diverse fields to show the implementation and application of our proposed OCA method for extracting insight from arbitrary cluster-shape. We analyzed datasets of (i) cartographic data of actual forest cover type; (ii) performance data of high performance computing (HPC) STREAM benchmark. We show that studying the deformation of topology preserved space is useful and efficient in detecting finer-grain pattern and relation among the stratified data. The innovation in our method is to show that local geometrical feature of clusters is important and can mathematically lead to novel and profounder insights from the data. In continue, we discuss the OCA parameter selection and evaluation method then we go into the analyses of datasets.

6.5.1 Parameter Selection

Our OCA method has only one parameter, which is n in performing the self-organizing map. The size of the map n is determined by calculating the number of neurons from the number of data points using $n \approx 5\sqrt{N}$, which is an integer close to the result of the right-hand side of the equation, and N is the number of observations [241].

6.5.2 Evaluation Quality of the Map

The important measure of the quality of the mapping is the topology preservation [242]. We calculate the topographic error, t_e , i.e. the proportion of all data vectors for which first and second Best Matching Units (BMU) are not adjacent units.

$$t_e = \frac{1}{N} \sum_{i=1}^N u(x_i) \quad (6.12)$$

where $u(x_i)$ is equal to 1 if first and second BMU are adjacent and 0 otherwise. So $t_e \in [0, 1]$, the map highly preserves topology when $t_e = 1$ or $t_e = 0$ poorly.

6.5.3 Identifying Spatial Patterns of Wilderness Sub-area.

The first application is the identification spatial patterns of wilderness sub-area. Identifying spatial patterns among potentially complex Geographical Information System (GIS) data in a consistent manner is a challenge in the field since sub-area can be small and have complex relationships. We show here that OCA can finely lead us to identify these spatial patterns by analyzing cluster-shape. We also identified interesting wilderness sub-area and their innate organization structure that may be important for geoscientists.

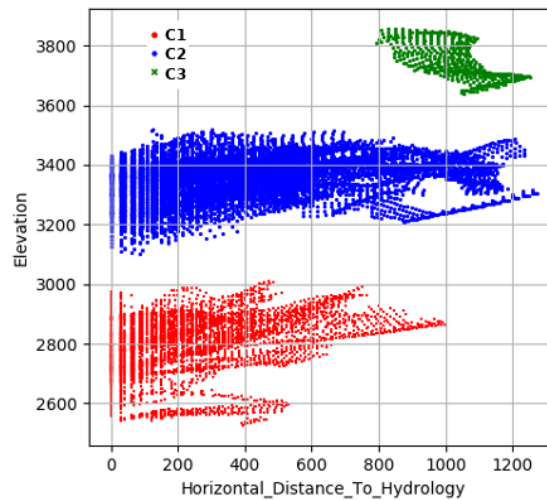


Figure 6.3: Comanche Peak Wilderness Area, visualizations of the clustering result (DBSCAN $eps = 0.15$, $minPts = 30$).

6. New Method for Extracting Insights from the Shape of Cluster

We use Covertypes¹ dataset (580,112 variable pairs, 17 numerical GIS variables such as elevation, slope, aspect, distance to hydrology, and etc.), to demonstrate fine-grain spatial patterns that can be identified among GIS data by using our OCA approach. Instances in the dataset are drawn from four different wilderness areas from the Roosevelt National Forest in north Colorado: Rawah, Neota, Comanche Peak and Cache la Poudre, which are covered with seven different tree species. The organization components derived from the cover-type data of species vary from area to area, with some areas-species having particular pattern. We took the Comanche Peak areas as a benchmark (253,364 data points), which would tend to be more typical of the overall dataset, while this area would probably have Aspen as their primary major tree species, followed by Krummholz [243].

By applying Robust Independent Feature Selection [14] method, we find that "elevation" and "horizontal_distance_to_hydrology" are an excellent active feature subset for categorizing wilderness sub-area, since most of tree species in the studied wilderness areas grow within specific ranges of altitudes and available moisture in a given cell. Then we apply the DBSCAN on this area-species by tuned hyper-parameters $eps = 0.15$ and $minPts = 30$ and using the active features. As a result, we stratify three distinct clusters in complex arbitrary shape, see Fig.6.3. We detect tendril cluster C1 in the lower part of that altitudinal zone. In contrast, we identify a small heterogeneous cluster C3 in highest elevation and an elongated cluster C2 with three tendrils.

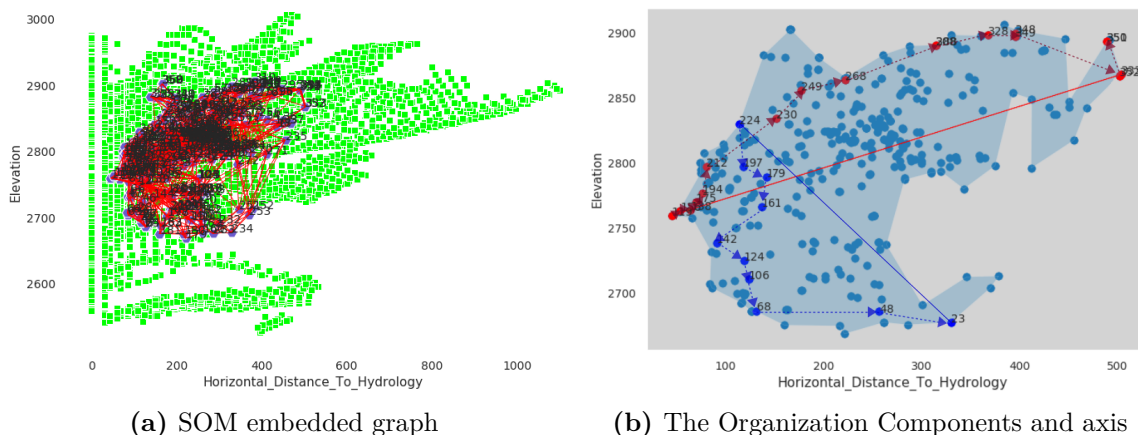


Figure 6.4: Application of OCA to Covertypes dataset cluster C1; (a) shows the data points and the SOM embedded graph.(b) The red and blue dashed arrows represent first and second Organization Component, respectively.

We applied OCA in each cluster based on GIS data. In order to randomize the experiments, we conducted 10 OCA on each chosen cluster. Then, for each cluster, the

¹<https://archive.ics.uci.edu/ml/datasets/covertypes>

6. New Method for Extracting Insights from the Shape of Cluster

average features influence as well as the standard deviations that have been computed over all analysis. OCA achieved to the 98%, 95%, and 84% topology preservation in average for cluster $C1$, $C2$ and $C3$ respectively by less than 300 iteration in average.

In Fig.6.4, as an example, we show plots of the organization components that we have derived from cluster $C1$. Plot (a) shows the embedded graph that is computed by our OCA method. The resulting graph has a structure shaped like a horizontal letter C. As shown in plot (b), our OCA method identified two spatial patterns in data. For first OC (red), the relationships are mainly aligned with increasing the active feature values where there is a long connected path, but in other OC (blue), the networks show a significantly decremental short path. Table.6.1 presents the Directional Sequence Similarity that is computed by OCA for each cluster. In case of cluster $C1$, this very high degree of DSS (69%) is evident with "Slope" in first OC. There are geological issues that could explain such spatial pattern. The associated data points to this OC are mainly belonged to Catamount soil family that is geomorphically positioned in mountain slopes in nature. We also determined that the horizontal distance changes to roadway can describe the spatial pattern associated with the second OC. The associated data points to this OC are mainly belonged to Bullwark soil family that is geomorphically positioned in mountain faceted spurs. Note that the faceted spurs usually ends up to flat area which is appropriate place for making road. In case of cluster $C2$, we can see euclidean distance to hydrology identified as most influencing feature in first two organization components, with approximately 92% and 70% DSS. Interestingly, the rest of the features mostly do not show significant Directional Sequence Similarity with the active feature set. In cluster $C3$ case, the cluster is shaped with the natural fire lines as most influencing feature with DSS 76%.

In order to verify our result, we carried out manually map visual study via ArcGIS and UCDAVIS ². On the map, we just applied the GIS data which have been computed as most influencing features by our OCA method. As a result, we could easily identify these sub-areas spatial pattern on map. For example, once we recognized a spatial pattern in the hydrological map which is identical to shape of cluster $C2$, the OCA of the sub-area reveals its organization structure significantly similar to euclidean distance to hydrology. We figure out another instance that the associated sub-area to $C3$ is a flat (high hill-shade noon) near to the Comanche peak and it has not been touched with natural fires.

Moreover, we find that a PCA analysis of the same categorized data was not able to detect the indicator to detect spatial pattern. For example, in Fig.6.5 we presented the cluster $C1$ PCA result as contribution bi-plot. The blue vectors are presenting the coordinates of the active features that are calculated as the correlation

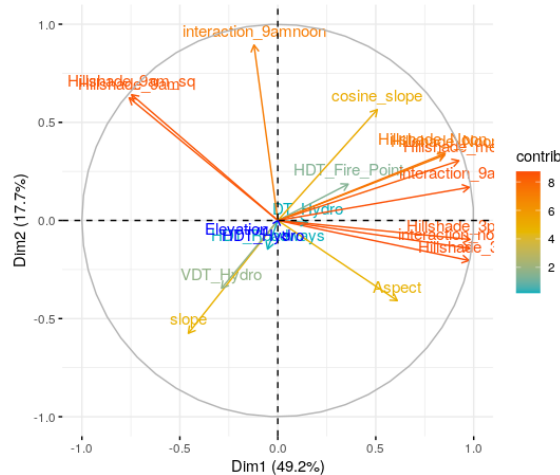
²<https://casoilresource.lawr.ucdavis.edu/see/>

6. New Method for Extracting Insights from the Shape of Cluster

Table 6.1: Application of OCA to Comanche Peak Wilderness sub-areas, the features influencing in the first and second organization components for each cluster.

Feature	Cluster C1		Cluster C2		Cluster C3	
	OC1	OC2	OC1	OC2	OC1	OC2
Aspect	0.196	0.248	0.094	0.135	0.020	0.010
Slope	0.688	0.162	0.069	0.238	0.259	0.077
VDT_Hydro	0.247	0.305	0.202	0.131	0.148	0.095
HDT_Road	0.218	0.462	0.060	0.219	0.184	0.138
Hillshade_9am	0.169	0.206	0.049	0.216	0.192	0.120
Hillshade_Noon	0.176	0.219	0.059	0.182	0.361	0.446
Hillshade_3pm	0.183	0.253	0.09	0.149	0.105	0.067
HDT_Fire_Point	0.196	0.206	0.061	0.100	0.753	0.085
Hillshade_mean	0.175	0.222	0.060	0.182	0.191	0.144
Hillshade_9am_sq	0.171	0.215	0.053	0.203	0.353	0.246
DT_Hydro	0.164	0.201	0.944	0.697	0.293	0.358
Hillshade_Noon_sq	0.174	0.221	0.059	0.187	0.212	0.171
Hillshade_3pm_sq	0.177	0.238	0.072	0.167	0.148	0.103
cosine_slope	0.172	0.226	0.058	0.191	0.235	0.208
Interac_9amnoon	0.173	0.202	0.049	0.203	0.268	0.267
Interac_noon3pm	0.183	0.248	0.089	0.145	0.092	0.056
Interac_9am3pm	0.179	0.234	0.07	0.162	0.138	0.089

between them and the principal components. As expected, there is a very weak linear correlation between them that means a PCA analysis of the same data was not able to identify the spatial connectivity.



(a) PCA bi-plot

Figure 6.5: Application of PCA to Comanche Peak Wilderness detected sub-areas C1. The larger the value of the contribution is, the more the feature contributes to the components.

In summary, we have identified any wilderness sub-area occurring consistently aligned with some spatial pattern but non-linearly. We note that these spatial

6. New Method for Extracting Insights from the Shape of Cluster

patterns are easily indicated by our methods because of the topology preservation property enjoyed by our approach. Moreover, we show that classical multivariate analysis approaches such as PCA, cannot easily detect these relevant indicators because by their nature they end up linearly separating points in the dataset that are in fact topologically close.

6.5.4 Diagnosing Performance Bottleneck in HPC Applications.

The next dataset we studied is a dataset that includes various Performance Hardware Counters³ (HWC) values in the HPC STREAM⁴ benchmark. Performance hardware counters values are unique metrics to understand the behavior of the application in a given hardware. Hardware counters are available in almost all modern processors, and count micro-architectural events such as L1, L2, L3: Levels of cache misses, MSP: Conditional branch instructions mispredicted, INS: Total instructions executed, and etc. Moreover, we drive a performance metric "Overlapping Index" (BOI) to indicate proportion of shared resources on-chip. The STREAM benchmark is a state-of-art HPC benchmark designed to measure sustainable memory bandwidth (in MB/s) and a corresponding computation rate for four simple vector kernels (Copy, Scale, Add and Triad). We executed STREAM application on the MareNostrum⁵ where *OMP_threads_number* = 40 and *Loop_size* = 9M to collect the dataset (4264 variable pairs, 9 numerical HWC variables) by specified interval sampling mechanism⁶. We mathematically diagnose patterns in these datasets that characterize the application performance losses by applying our OCA approach. Note that OCA extracted these deep insights without requiring the expertise to study the huge amount of information manually and visually.

In [8], they propose the Completed Instructions (INS) combined with Instructions Per Cycle (IPC) as an appropriate active feature subset. This combination focuses the clustering on the "performance view" of the application. Then, we applied DBSCAN algorithms to the extracted performance data by tuned hyper-parameters and using the active features in order to determine the application structure. Fig.6.6 shows the detected clusters (application phases). As a result, we stratified four distinct clusters in elongated shape. One can then ask the question if each cluster represents a distinct phase of application why they show heterogeneous performance behavior. To answer this question, we applied the OCA to detect the HPC systems bottleneck that can describe the variability among stratified data.

³<https://icl.utk.edu/papi/>

⁴<http://www.cs.virginia.edu/stream/>

⁵<https://www.bsc.es/marenostrum/marenostrum>

⁶<https://tools.bsc.es/extrae>

6. New Method for Extracting Insights from the Shape of Cluster

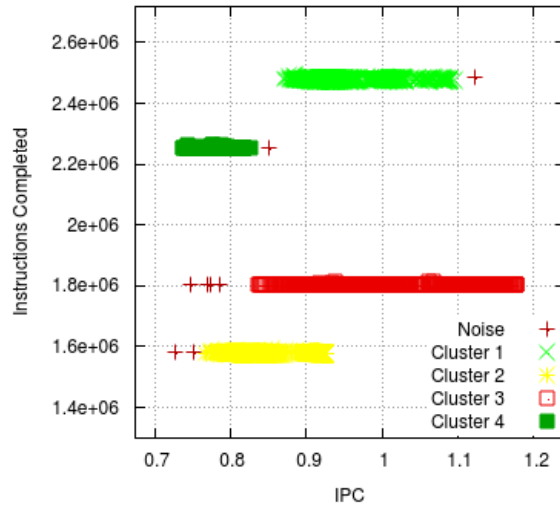


Figure 6.6: Performance data extracted from STREAM benchmark execution ($OMP_threads_number = 40$, $Loop_size = 9M$), visualizations of the clustering result (DBSCAN $eps = 0.015$, $minPts = 6$).

In this section, we report the results of the Triad operation (e.g., Cluster1 in Fig.6.6), since it is the most complex scenario and is highly relevant to kernels used in HPC applications. In order to randomize the experiments, we conducted 10 OCA on Triad associated cluster and computed the average features influence as well as the standard deviations over all analysis. In all conducted experiments, OCA achieves to 94% topology preservation in average, which shows strong mapping quality.

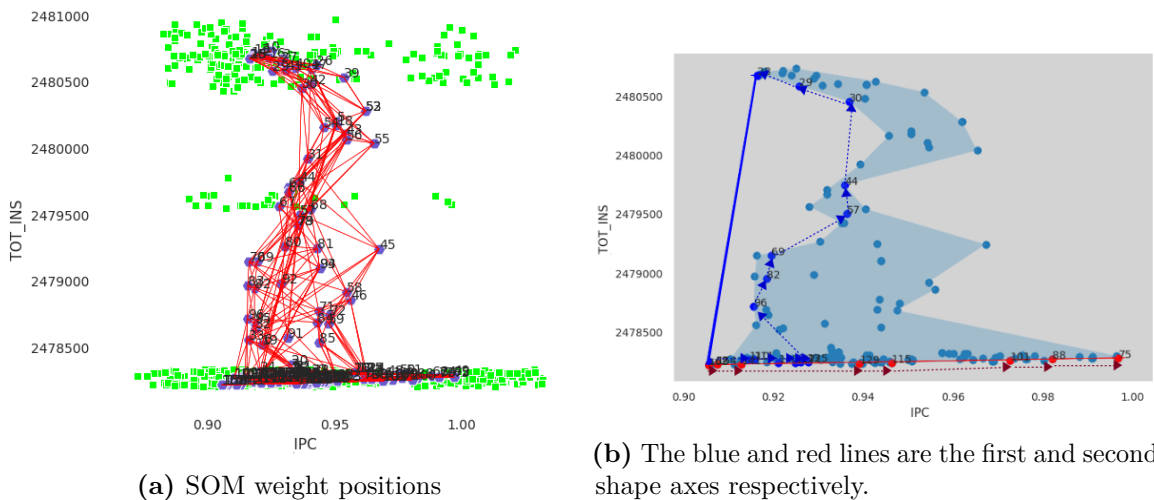


Figure 6.7: Application of OCA to STREAM dataset cluster C1; (a) shows the locations of the data points and the weight vectors.(b) The blue points are representing the SOM neurons, while the red and blue dashed arrows represent first and second Organization component, respectively.

Fig.6.7 shows plots of the organization components that we have identified in *cluster1* of previously aforementioned example. Plot (a) shows the embedded graph

6. New Method for Extracting Insights from the Shape of Cluster

that is computed by our OCA method. Although all data points are representatives of the same kernel, they can be categorized into three distinct sub-clusters that do not detected by DBSCAN due to the fact that we select the ε value to identify the main application trends. Note that the bottom sub-cluster includes the majority of the data points. As shown in (b), our OCA method identified two sub-structures. We identify that the first organization component indicates the main HPC application performance behavior and the second one presents the reason of the inter-cluster stratification. From Table.6.2, we diagnose that the magnitude of shared on-chip resources (BOI) can describe approximately 70% of the performance losses, since each thread can only use a fraction of the shared resources at specific moment. Furthermore, we identify the performance effect of L1\L2, L1 and L2 miss ratio have become the main concern when following the OC2 trajectory path, it is likely that approximately 50% of the performance problem is the L1 and L2 capacity. Also, we detect three sub-groups that represent distinctive level of L1 and L2 miss ratio.

Table 6.2: Application of OCA to STREAM data set cluster C1, the features influencing in the first and second OC.

	L1	L2	L3	L1\L2	L2\L3	MSP	BOI
OC1	0.223	0.220	0.159	0.227	0.222	0.051	0.693
OC2	0.492	0.500	0.086	0.483	0.134	0.015	0.189

In advanced experiment, we executed STREAM application with various combination of $OMP_NUM_THREAD \in \{1, 2, 4, 8, 16, 24, 32, 40, 48\}$ and $Loop_size \in [10k, 89M]$ to collect 134 datasets (1221 to 11,600 variable pairs, 9 numerical HWC variables). We conducted the prior process on each obtained dataset to diagnose the HPC system bottlenecks.

Fig.6.8, presents the contour plots of the mean value of IPC and INS (active subspace), versus the $\log(\text{loop size})$ and the $OMP_threads_number$ and Fig.6.9 shows contour plots of illustrative feature’s Directional Sequence Similarity with the major organization component. As shown in the plot 6.8.a, the application roughly shows highest performance by the small number of threads and it has high performance in the area under the bell-shaped component as well. We detect significantly similar pattern in the plot 6.9.g which identifies the performance of application mainly influenced by the magnitude of shared on-chip resources. We also identify a minor difference between two components in the right tail. It is caused by the imbalanced thread distribution between sockets that increases the IPC mean. For example, in $OMP_NUM_THREAD = 16$ and $\log(Loop_size) = 18$ case only two threads are assigned to the second socket; fourteen threads to the first one. From the plots 6.9.(a ~ e), we can conclude that the bottleneck of the small, medium and big

6. New Method for Extracting Insights from the Shape of Cluster

problem sizes is L1, L2 and L3 misses ratio respectively. The most remarkable aspect of the plot 6.9.a, see the yellow area in the small loop size, is that the exponential relationship between loop size and OMP_number_threads. In the same way, we recognize the similar pattern in the performance of application (IPC), you can see the lightest orange area in the small loop size in plot 6.8.a. Although we identify L1 miss rates as a main bottleneck of small problem size, the performance of application has still been acceptable, probably, in consideration of the relatively low latency of L1 cache. It would be worth mentioning that, on the very small loop size, the performance of application is reduced quickly by increasing the number of threads, possibly, due to the fact that parallelism overhead is too remarkable in case of very small vector size, as it can be seen in the right side of the plot 6.9.f. As it can be seen in plot 6.8.b, there is roughly a linear relationship between INS and *Loop_size* due to the vector size. Although the INS does not illustrate a performance issue, it helps use to segregate main application trends perfectly.

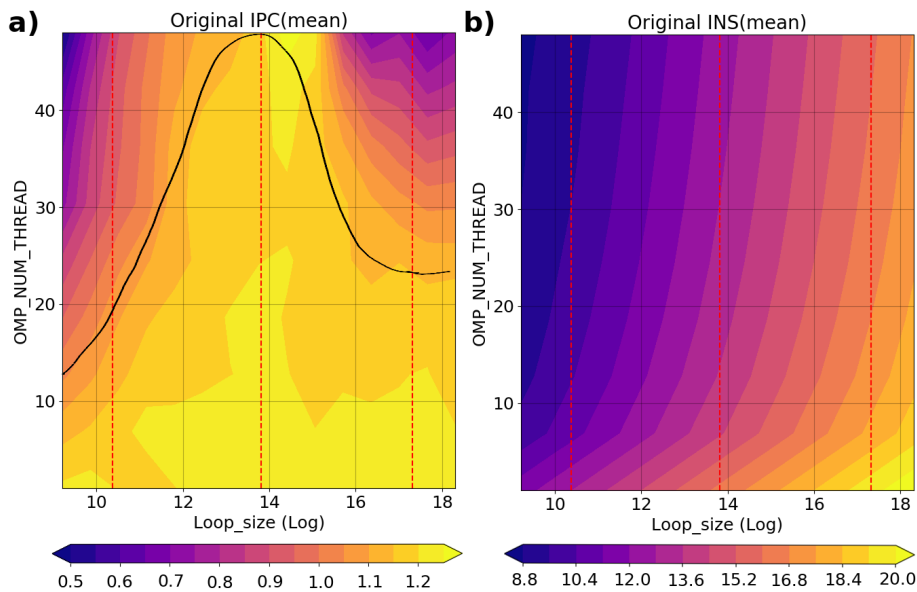


Figure 6.8: STREAM benchmark, the plots are shown the contour plots of the mean value of IPC and INS, versus the log(loop size) and the OMP_threads_number.

In order to verify the insight that extracted by the OCA, we performed manually the HPC performance analysis with PARAVR ⁷ toolkit and we identified the same performance bottleneck as well. However, it is a manual approach.

In summary, our Organization Component Analysis diagnoses performance bottleneck of HPC applications automatically rather than the visual approach that is too intuitive and laborious. In case of STREAM benchmark, we identified that

⁷<https://tools.bsc.es/paraver>

6. New Method for Extracting Insights from the Shape of Cluster

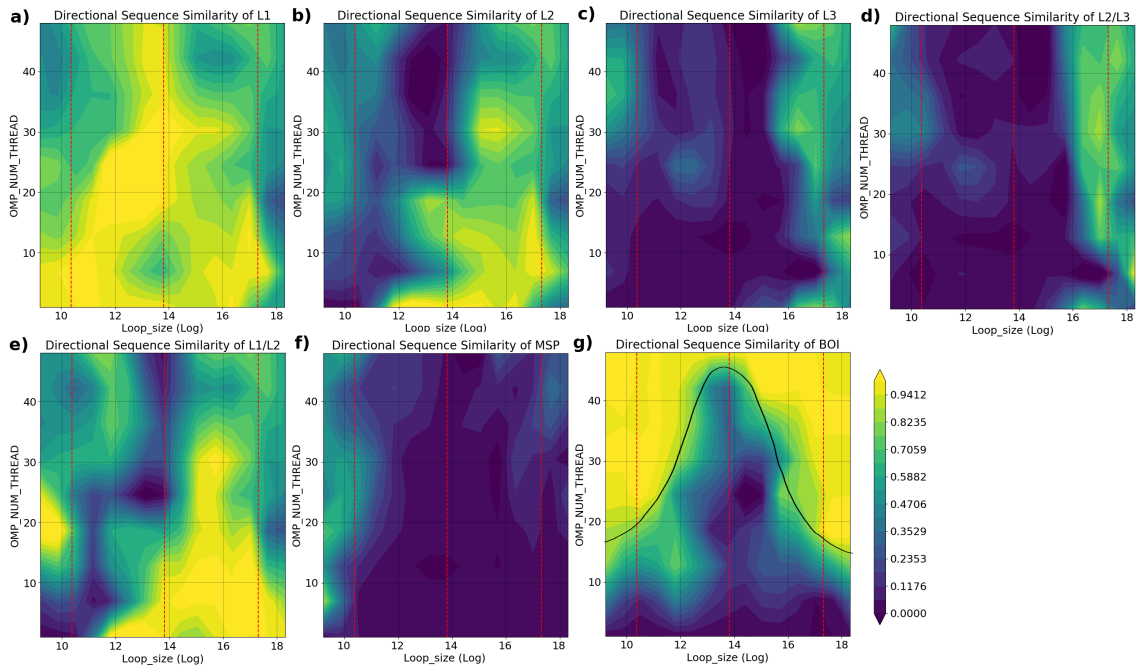


Figure 6.9: STREAM benchmark, the plots present the contour plots of illustrative feature’s Directional Sequence Similarity with the major organization component, versus the log(loop size) and the OMP_threads_number. The red dashed lines show the threshold of three hierarchical levels of caches (32kB, 1MB and 33MB) receptively.

higher magnitude of shared resources on-chip will ruin the application performance dramatically. Meanwhile the number of cache misses in the higher level of cache hierarchic will be the performance bottleneck by increasing the loop size. In general, the OCA can be a canonical approach to automatically detect the HPC application performance bottleneck among complex performance data without expert human visual analysis, which can broadly be applied to any HPC application.

7

Enhanced Cluster Identification and Interpretation Pipeline

Contents

7.1	Background and Motivation	120
7.1.1	The Limitation of the DBSCAN	122
7.2	ECII Pipeline Architecture	125
7.2.1	Hyperparameter Optimization	126
7.2.2	Evaluation Metrics	128
7.2.3	Components Interaction	129
7.3	Practical Uses to Application Analysis	133
7.3.1	GROMACS	133
7.3.2	Computation Bursts and Enhanced Cluster Analysis	133
7.3.3	Application Analyses	134

In this chapter, we present our Enhanced Cluster Identification and Interpretation (ECII) Pipeline, a new approach to automatizing the clustering process. It overcomes the limitations of DBSCAN to tune its hyperparameters parameters, and it helps non-expert users to extract fine-grain insight from the clustering results. Furthermore, we demonstrate the usefulness of the described pipeline by evaluating the performance of the state-of-art parallel application.

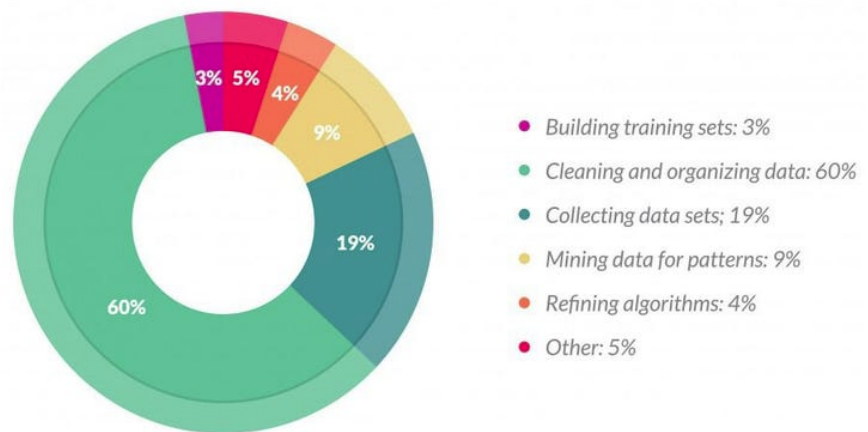
7.1 Background and Motivation

Traditional machine learning processes are inherently time-consuming and dependent on human intervention and expertise. The theorem of “no free lunches” [244] also

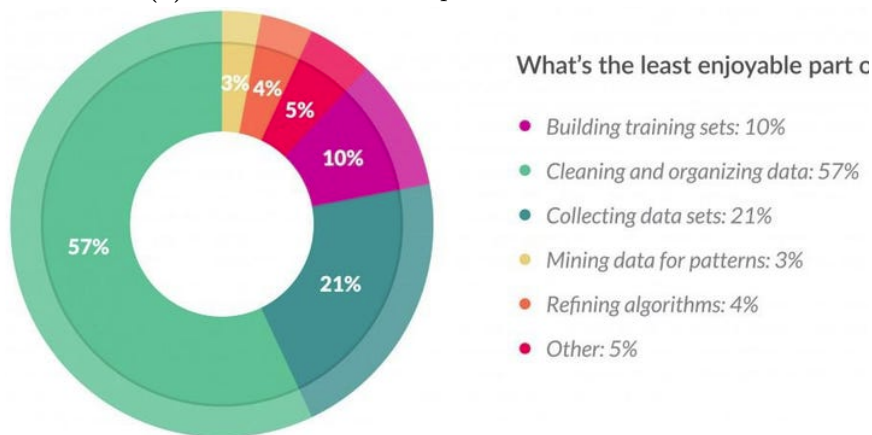
7. Enhanced Cluster Identification and Interpretation Pipeline

implies that the only scenario when an approach outperforms another is when it is customized to the specific problem on hand. This means that the model development invariably has to go through the pains of data preparation, feature selection, feature engineering, model training, evaluation, and tuning. Every time a new dataset is encountered or a new problem arises.

In a report published by Forbes [245], data cleaning and preparation occupies 60% of a data scientist's workload, see Figure 7.1a. Since 57% of data scientists regard cleaning and organizing data as the least enjoyable part of their work, see Figure 7.1b. On the other hand, tasks such as data preparation and exploring multiple models to select the optimal one can be easily automated using ML pipeline frameworks.



(a) How data scientists spend most of their time.



(b) What data scientists enjoy the least.

Figure 7.1: The result of a survey of how data scientists spend their time [245]

Machine Learning Pipeline aims to automate and accelerate the process of building ML models and particularly clustering models in this work. It enables us to provide the unlabeled dataset as input and receive an optimized clusters structure of the dataset as output. This functionality enables users to generate profound insights by leveraging the capabilities of the Machine Learning Pipeline.

7. Enhanced Cluster Identification and Interpretation Pipeline

ML pipeline enables professionals from various domains to leverage the benefits of data science and Machine Learning. It accelerates processes, reduces errors and costs, and provides greater accuracy of results by training multiple high-performing models. It cuts down time spent on iterative tasks concerning model development. ML pipeline channelizes more of experts' time and resources into the model selection and model tracking so that they can deal with more complex problems. Furthermore, these tools can help developers to build scalable models with pipeline output acceleration.

7.1.1 The Limitation of the DBSCAN

Although DBSCAN is the most popular and widely used density-based clustering algorithm that can detect the clusters of the arbitrary shapes and sizes in large databases, it still has limitations. Three major drawbacks can be associated with the DBSCAN algorithm which we propose our machine learning pipeline to overcome.

7.1.1.1 Hyperparameter Selection

Although DBSCAN has significant advantages in clustering, it also has the same defects as other clustering algorithms, that is, the clustering performance depends on the parameter settings. DBSCAN has two parameters *eps* and *minPts*, and a given combination of them could lead to different cluster identification results. Choosing the wrong values for these two parameters can lead to all points to be in one single cluster or distinguish as noise. If the dataset and features are not so well understood by a domain expert then, setting up the appropriate *eps* and *minPts* could be tricky and may need comparisons for several iterations with different values of *eps* and *minPts*. The ECII pipeline helps us to automatically tune the DBSCAN hyperparameter to obtain the optimal clustering result.

In different datasets, the optimal clustering results of DBSCAN will have different values of the parameters *minPts* and *eps*. Even in the same dataset, *minPts* takes different values, and the optimal value of *eps* is quite different. In addition, there are no theoretical guidance parameters for setting *minPts* and *eps*, which leads to the selection of reasonable DBSCAN parameters. The reasonable DBSCAN parameters completely depend on personal experience and a large number of experimental trials. Since the clustering result of DBSCAN is sensitive to parameters, the parameters *minPts* and *eps* must be set reasonably in the application, which limits the extensive use of DBSCAN to some extent.

For this problem, many scholars have done some research on the direction of the parameters setting of DBSCAN. In [246] and [247], the parameter *eps* is automatically determined by using the k-dist list. In reference [248], the authors propose a hierarchical

7. Enhanced Cluster Identification and Interpretation Pipeline

adaptive alternating optimization method to find the optimal parameter combination of DBSCAN. In [249], a normalized density list is generated by evaluating the local density of the dataset by using the Affinity Propagation algorithm, and then the density list is combined to determine the parameters of the DBSCAN. In [250], the histogram equalization is applied to the pairwise similarity matrix of input data, and then the optimal parameter combination of DBSCAN is determined by dominant sets (DSets). In [251], the binary differential evolution algorithm is used to optimize the optimal combination parameters $minPts$ and eps . These methods have promoted the development and application of DBSCAN to some extent. In this dissertation, a new DBSCAN parameters optimization approach based on the grid search optimization algorithm is proposed, which is not for finding a value of the parameters in the optimal clustering but for finding the interval of the parameters to search for their optimal configuration. Our method enables DBSCAN to select a more reasonable value of eps and $minPts$ from the optimal range when clustering.

Some particular hybrid-DBSCAN methods exist to solve difficulties in finding appropriate input parameters. Darong and Peng [252] combine the grid partition technique and DBSCAN to automatically generate input parameters. The efficiency of this method has not been evaluated against noise and various data sets with different densities. This method also requires input parameters for grid partitioning. Smiiti and Elouedi [253] combine Gaussian-Means (GM) and DBSCAN algorithm to determine the input parameters in DBSCAN. However, GM provides the circular cluster shape not the density-based clusters, and it is not strong against noise (outlier). It still needs input parameter for the Gaussian distribution. These existing algorithms and techniques still have their own drawbacks and limitations which lead to a bad clustering.

Recently, some researches have combined clustering algorithms with optimization and meta-heuristic algorithms to improve the results of clustering. For instance, Simulated Annealing [254], Particle Swarm Optimization [251, 255–257], Tabu Search [258, 259], Harmony Search [260–262], Bees algorithm [263–265], and Ant Colony Optimization [266, 267]. However, there is no particular research to solve the problem of automatically choosing input parameters in DBSCAN algorithm. In this thesis, we combine the DBSCAN algorithm with a new optimization approach to choose well-suited DBSCAN input parameters.

Since a slightly different setting (changing $minPts$ and eps values) in DBSCAN may lead to total different clusters of a data set [268], an optimization procedure would fit the requirements (finding the optimal combination of $minPts$ and eps) for a good clustering. Because an optimization algorithm is related to an optimal choice of process decisions that satisfy definite constraints and make an optimization criterion (performance or cost index) maximize or minimize [269, 270]. Since the eps parameter

7. Enhanced Cluster Identification and Interpretation Pipeline

can largely degrade the efficiency of the DBSCAN algorithm [271], the combination of the analytical ways for estimating the parameters is employed. Our method can create more diverse configuration of the parameters until an appropriate combination of $minPts$ and eps values be selected. We decided to use an exhaustive grid search as a preliminary optimization approach to establish preliminary ECII pipeline.

7.1.1.2 Varied Density

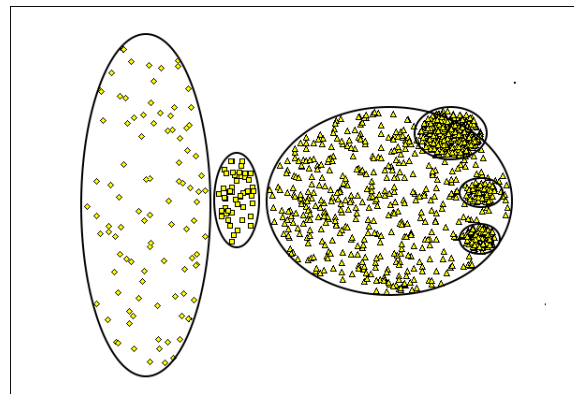
The second problem is the inability of DBSCAN to correctly detect clusters when the density varies across the feature space. This problem is directly related to the use of a single eps and $minPts$ value. In Figure 7.2 we present a practical example. Figure 7.2a shows the original dataset clustering structure where each cluster is specified by an ellipse. Figure 7.2b shows the identified clusters by relatively big eps where compact clouds of points on the right side of the plot are not identified. To correctly detect these clouds as different clusters we require using a smaller eps value. On the other hand, Figure 7.2b shows the identified clusters by relatively smaller eps . Although these compact data clouds are accordingly identified by DBSCAN, the sparse cluster on the left side of the plot is distinguished as noise. Therefore, we equipped our ECII pipeline with a new transformation technique to homogenize cluster densities while preserving the topological structure of the dataset. We describe our Feature Space Curvature Map (FSCM) technique with details in chapter 5.

7.1.1.3 Extracting Insight from the Clustering Result

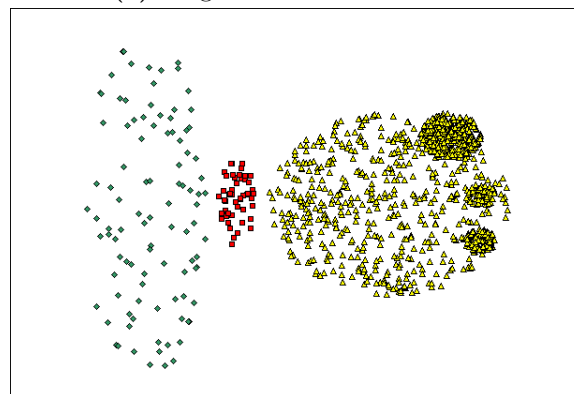
In the end, after conducting the Clustering algorithm on the prepared data, interpreting the result of the cluster analysis is the most crucial step. Distillation insight, the last stage of the cluster analysis process, tries to find those hidden correlations among data points, which are now formed into clusters. However, the machine learning experts should perform this interpretation, and it is more challenging when there are multidimensional clusters. Accordingly, in our ECII pipeline, we apply a novel topological-based method to study potentially complex high-dimensional categorized data by quantifying their shapes and extracting fine-grain insights about them to interpret the clustering result. We describe our Organization Components Analysis (OCA) technique with details in chapter 6.

Moreover, the validated cluster models should be interpreted in light of the included dataset context. Therefore, a model can have a very good fit with the data, but the result can be uninformative. Feature selection is one of the key ways to avoid models with no practical application value. In our pipeline, we embed a new feature selection technique in the ECII pipeline where selected features are immune to the noise. We describe our Robust Independent Feature Selection (RIFC) technique with details in chapter 4.

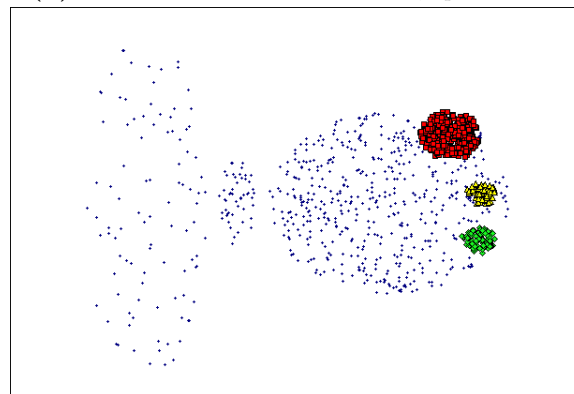
7. Enhanced Cluster Identification and Interpretation Pipeline



(a) Original data stratification.



(b) DBSCAN result $minPts = 4$, $eps = 9.92$.



(c) DBSCAN result $minPts = 4$, $eps = 9.75$. The blue + points are noise.

Figure 7.2: Example of dataset of varies density across the feature space.

7.2 ECII Pipeline Architecture

We use our ECII pipeline to automate clustering analysis workflow and implement AutoML solutions. The pipeline starts with ingesting raw data, which is then engineered to suit algorithm and domain requirements by leveraging feature selection and transformation. The clustering model is then trained on this data and validated by hyperparameter tuning. The best-performing model is then deployed and used

for insight extraction.

7.2.1 Hyperparameter Optimization

ECII hyperparameter optimization process has three components: (1) a search space, (2) a search method, and (3) an evaluation method. We need to decide on a set of hyperparameter values that we want to investigate, and then we use our DBSCAN model to calculate the corresponding evaluation method. Finally, we can choose the optimum ($eps, minPts$) combination as the one that maximizes the mean Silhouette. We describe each component in turn below.

7.2.1.1 Search Space

An optimization procedure involves defining a search space. This can be thought of geometrically as an n -dimensional volume, where each hyperparameter represents a different dimension. The scale of each dimension is the value that a related hyperparameter may take on. In our case, a point in the search space is a vector with a specific value for the eps and $minPts$.

Lets X indicate a dataset of n data points $X = (x_1, x_2, \dots, x_n)$ where $x_i \in R^m$.

minPts The purpose of $minPts$ is to smooth the density estimates. Because a cluster is a maximal set of density-connected points to choose smaller values when the expected number of detections in a cluster is unknown. However, smaller values make the DBSCAN algorithm more susceptible to noise. Conclusively, the $minPts$ value should be set using domain knowledge and familiarity with the dataset. There is no automatic way to determine the $minPts$ value for DBSCAN in the literature. Hence, we use the following general guideline [10, 272] to mathematically define the $minPts$ search space:

- Generally, set $minPts \geq 2m$ where m is the number of features in X .
- Increasing the $minPts$ can often improve the clustering results for datasets that have one or more of the following properties:
 - many noise points
 - large number of points n

Due to the first role, we set the lower bound of the search space with the $B_l = 2m$.

Accordingly, to compute the upper bound, first of all, we use the average Mahalanobis Distance ($\overline{D_M}$) [273] of the dataset to measure the noise in dataset:

$$\overline{D_M} = \frac{1}{n} \sum_{i=1}^n D_M(x_i) \quad (7.1)$$

7. Enhanced Cluster Identification and Interpretation Pipeline

Where $D_M(x_i)$ is the Mahalanobis Distance of the data point $x_i \in R^m$ from the dataset X with mean $\mu = [\mu_1, \mu_2, \mu_3, \dots, \mu_m]$ and (non-singular) covariance matrix S which is defined as:

$$D_M(x_i) = \sqrt{(x_i - \mu)^T S^{-1} (x_i - \mu)} \quad (7.2)$$

It is a multi-dimensional generalization of the idea of measuring how many standard deviations away x_i is from the mean of dataset. Thus, we take to account the $\overline{D_M}$ as a noise measure.

Then, due to the second set of roles, we compute the upper bound as follows:

$$B_u = \max(2m + \overline{D_M} \cdot \log(n), \text{ApplicationTask}) \quad (7.3)$$

As a result, we define the *minPts* search space as follow:

$$SS_{minPts} = [B_l, B_u] \quad (7.4)$$

Epsilon (eps/ϵ) We can subsequently determine the *eps* search space. We use the leveraging of the elbow technique [81] to automatically determine the search space of the *eps* value. We calculate the average distance between each point and its k nearest neighbors, where $k \in \{B_l, B_u\}$, see Equation 7.4. Next, we plot the average k-distances in ascending order on a k-distance graph for each k individually. Then we use the Kneedle [83] technique to find the relatively optimal values for *eps* at the point where each graph has the maximum curvature, see section 3.1.3.1. As a result, we can define the *eps* search space as follow:

$$SS_\epsilon = [Kneedle(B_l), Kneedle(B_u)] \quad (7.5)$$

7.2.1.2 Search Method

We use the grid search to find the optimal hyperparameter configuration over the previously defined search space since it is simple to implement, its parallelization is trivial, and it is reliable on low dimensional spaces (e.g., 1d, 2d). Grid Search guarantees the detection of the best hyperparameters.

With this technique, we simply build a grid with each possible combination of all the hyperparameter values provided, calculating the score (mean Silhouette) of each DBSCAN model, to evaluate it, and then selecting the model that gives the best results.

First of all, we cast the problem as an optimization problem:

$$\lambda^* = \operatorname{argmax}_{\lambda \in \Lambda} \Psi(\lambda) \quad (7.6)$$

7. Enhanced Cluster Identification and Interpretation Pipeline

Where Λ denotes all hyperparameters configuration on previously defined search space, and Ψ is the hyperparameter response function for $DBSCAN_i$ which is the one obtained after choosing λ_i . We have to select two parameters, that is, $\lambda_i = (minPts_i, \epsilon_i)$ where $\forall minPts_i \in \{\mathbb{Z} \wedge SS_{minPts}\}$ and $\forall \epsilon_i \in \{\mathbb{R} \wedge SS_\epsilon\}$.

We call $\Psi(\lambda)$ the response surface over $\lambda \in \Lambda$ since it is 2D search space. Then we choose the number (S) of trial points $\Lambda = \{\lambda_1, \dots, \lambda_S\}$, to evaluate $\Psi(\lambda)$ for each one, and return the λ_i that worked the best as λ^* . This strategy is made explicit by Equation 7.6.

To choose the set of trials Λ , we use a combination of grid search and manual search, as well as scikits.learn¹ machine learning software packages. As a result, the grid search requires that we choose a set of values for each variable $minPts$ and ϵ . For the $minPts$, we choose each integer value belongs to the SS_{minPts} . For the ϵ , we slice the entire SS_ϵ interval into ξ sub-interval where ξ is manually set, and $\xi \geq 10$ works well.

Furthermore, the grid search can get extremely time consuming if the number of possible hyperparameters is large. Therefore, after trying a range of values, we can fine-tune by taking the best-performing hyperparameters and starting a new search centered on them.

7.2.1.3 Evaluation Method

The grid search would train the DBSCAN model with each hyperparameter configuration and output the configuration that achieved the highest score in the evaluation method defined in Equation 7.6.

7.2.2 Evaluation Metrics

We use the computationally abridged version of Silhouette [274] to evaluate the clustering result. Our version of Silhouette adopts a similar approach as that of the original Silhouette, but abridged the distance of a data point to a cluster from the average distance of x_i to all (other) data points in a cluster to the distance to the centroid of the cluster as follows:

$$a^*(x_i) = d_E(x_i, C_h) \quad (7.7)$$

$$b^*(x_i) = \min_{h \neq l} (d_E(x_i, C_l)) \quad (7.8)$$

And the abridged Silhouette value for a single data point $SI(x_i)$ is defined as:

¹<http://scikit-learn.sourceforge.net>

7. Enhanced Cluster Identification and Interpretation Pipeline

$$SI(x_i) = \frac{b^*(x_i) - a^*(x_i)}{\max^*(b(x_i), a^*(x_i))} \quad (7.9)$$

In the same way, the abridged Silhouette value for a full clustering \overline{SI} is defined as:

$$\overline{SI} = \frac{1}{n} \sum_{i \in X} SI(x_i) \quad (7.10)$$

The abridged Silhouette value also ranges in $[-1, 1]$. -1 shows a very bad clustering, while 1 shows a perfect clustering. As a result, the overall complexity of the computation of Silhouette is estimated as $O(mn^2)$, while that of abridged Silhouette is estimated as $O(qmn)$. When the number of clusters q is much smaller than n , Silhouette that is much more computationally expensive than abridged Silhouette.

7.2.3 Components Interaction

Figure 7.3 shows the ECII pipeline architecture. Pipeline orchestration tools, including MLflow², Scikit-Learn³, Scikit-Optimize⁴, and Docker⁵, are the foundation for executing our tasks. Besides the orchestration tools, we need a data store to keep track of the intermediate pipeline results. The individual components communicate with the data store to receive their inputs, and they return the results to the data store. These results can then be inputs to the following tasks. ECII provides the layer that combines all of these tools, and it provides a variety of pipeline components to enhance the density-based cluster analysis. The following components are available:

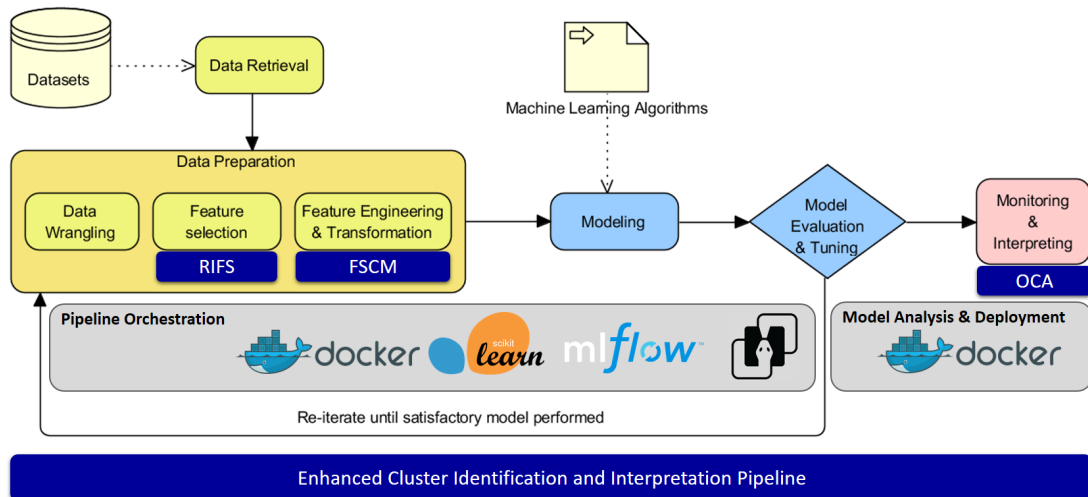


Figure 7.3: ECII pipeline architecture.

²<https://mlflow.org/>

³<https://scikit-learn.org/>

⁴<https://scikit-optimize.github.io/>

⁵<https://www.docker.com/>

7. Enhanced Cluster Identification and Interpretation Pipeline

Data Retrieval In this step of our pipeline, we read data files (e.g., csv pikelet xlsx) or parse the data from an external sources (e.g., prv) to ingest data into our machine learning pipelines. Before passing the ingested dataset to the next component, we convert the datasets into Pandas data-frame and then normalize the available data.

The data-retrieval component can ingest a few data structures, including comma-separated value files (CSVs), excel spread sheets, and python pikelet. Datasets for performance data or trace files are often stored in PRV files. ECII provides functionality to read and convert these files to data-frame data structures.

Data Preparation One major aspect of our ECII pipeline is focusing on consistent preprocessing. As shown in Figure 7.3, the data preparation takes place after data-retrieval. Scikit-learn Pipeline enables us to chain multiple estimator into one. This is useful as there is often a fixed sequence of steps in processing the data, for example data wrangling, feature selection, and transformation.

Data Wrangling The data we use to train our machine learning models can be provided in formats our models can't consume. For example, a feature we want to use to train our model is available only as *Yes* and *No* tags, and our machine learning model requires a numerical representation of these values (e.g., 1 and 0). In this step, we convert features into consistent numerical representations so that our machine learning model can be trained with the numerical representations of the features.

Feature Selection Next, we use our unsupervised approach for feature selection on noisy data, called Robust Independent Feature Selection (RIFS), see Chapter 4. Specifically, we choose a feature subset that contains most of the underlying information, using the same criteria as the Independent component analysis (ICA). Simultaneously, the noise is separated as an independent component. The isolation of representative noise samples is achieved using factor oblique rotation whereas noise identification is performed using factor pattern loading. Our unsupervised RIFS approach has an explicit mechanism for detaching and isolating the noise thus it can produce an optimal feature subset. On the other hand, almost all traditional unsupervised feature selection methods are not robust against the noise in samples.

Density Homogenization Then, we use our new Feature Space Curvature Map technique (FSCM) which is a parametric multilinear transformation method to homogenize cluster densities while preserving the topological structure of the dataset, see Chapter 5. The transformed clusters have approximately the same density while all inter-cluster regions become globally low-density. In our method, the feature space

7. Enhanced Cluster Identification and Interpretation Pipeline

is locally bent by dense data point concentrations the same way as stars bend the space-time dimensions in the Theory of Relativity. We present a new Gravitational Self-organizing Map to model the feature space curvature by plugging the concepts of gravity and fabric of space into the Self-organizing Map algorithm to mathematically describe the density structure of the data. To homogenize the cluster's density, we introduce a novel mapping mechanism to project the data from a non-Euclidean curved space to a new Euclidean flat space. Specifically, this mechanism transfers the basis vectors instead of the feature vectors to guarantee the continuity of the mapping function and optimize the computation cost of the algorithm. As a result, our method can efficiently and explicitly homogenize the density of any dataset globally to then apply existing clustering algorithms without modification.

Clustering Modeling Now that the data preprocessing step is complete and the data has been transformed into the format that our model requires, the next step in our pipeline is to train the model with the freshly transformed data.

This ensures that all the data needed by the model is present and that it has been reproducibly transformed into the features that the model requires. All of these are necessary because we don't want the pipeline to fail at our next step. We want to ensure that the training proceeds smoothly because it is often the most time-consuming part of the entire pipeline.

One very important feature of training a model in our pipeline is that the data preprocessing steps that we discussed in the previous section are saved along with the trained model weights. This is incredibly useful once our model is evaluated because it means that the preprocessing steps will always produce the features the model is expecting. Without this feature, it would be possible to update the data preprocessing steps without updating the model, and then the model would fail or the extracted insight would be based on the wrong data.

The initial cluster algorithm used was a K-means-like algorithm. As we explained in Chapter 3, K-means-like algorithms always suppose a Gaussian model of the data. However, the performance hardware counters data is not distributed following this Gaussian model so the results obtained were not satisfactory. Finally, we selected DBSCAN as the representative of density-based clustering due to its no assumption of the underlying model. Even hierarchical clustering also shares this property with density-based clustering, we discarded it due to the difficulty to manipulate dendrograms with a high number of points.

7. Enhanced Cluster Identification and Interpretation Pipeline

Model Tuning Hyperparameter tuning is an important part of achieving an accurate machine learning model. Depending on the use case, it may be something that we do during our initial experiments or it may be something we want to include in our pipeline as we did in our ECII pipeline.

We use the grid search as hyperparameter search approaches, see Section 7.2.1. In grid search, every combination of parameters is tried exhaustively.

Scikit-Optimize, or skopt for short, is an open-source Python library for performing optimization tasks. It offers efficient optimization algorithms, such as Grid search, and can be used to find the minimum or maximum of arbitrary cost functions. In addition to grid search, both of these packages support Bayesian search and the Hyperband algorithm. We use the leveraging of this library to tune our clustering models.

Importantly, the library provides support for tuning the hyperparameters of machine learning algorithms offered by the scikit-learn library, so-called hyperparameter optimization. As such, it offers an efficient alternative to less efficient hyperparameter optimization procedures.

Interpreting Each cluster has a unique shape that comes out of metrics on data, which can represent the organization of categorized data mathematically. Accordingly, we proposed a novel topological-based method to study potentially complex high-dimensional categorized data by quantifying their shapes and extracting fine-grain insights about them to interpret the clustering result. We introduce our Organization Component Analysis method for the automatic arbitrary cluster-shape study without an assumption about the data distribution. Our method explores a topology-preserving map of a data cluster manifold to extract the main organizational structure of a cluster by leveraging the self-organizing map technique. To do this, we represent the self-organizing map as a graph. We introduce organization components to geometrically describe the shape of clusters and their endogenous phenomena. Specifically, we propose an innovative way to measure the alignment between two sequences of momentum changes on the geodesic path over the embedded graph to quantify the extent to which the feature is related to a given component. As a result, we can describe variability among stratified data, correlated features in terms of the lower number of organization components.

Pipeline Orchestration We orchestrate our pipelines with Sklearn pipeline. Developing pipeline today requires so much more than writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each life-cycle stage create enormous complexity. We use Docker to simplify and accelerate the workflow. Sklearn pipeline with Docker containers enable us to use Docker images as

7. *Enhanced Cluster Identification and Interpretation Pipeline*

the execution environment for a single Stage or the entire Pipeline. Our orchestration approach takes care of the coordination between the pipeline components. Then, we use MLflow to make packing Sklearn models efficiently.

We will likely need to write our own Dockerfile if our model has dependencies that can't be included by MLflow's docker build script. The run script should start the MLflow model service on the artifact from the previous step.

7.3 Practical Uses to Application Analysis

In this section, we evaluate the performance of the GROMACS application executed on the MareNostrum⁶ 4 to demonstrate the value of our ECII framework and especially the FSCM mechanism presented in this thesis. Our framework can point out the nature of the fine-grain performance bottlenecks and their root causes. This analysis enables the analyst to understand the the computation structure and the characteristics of the application. Moreover, this information guides the analyst to apply the modifications to correct the inefficiencies and improve the overall performance.

7.3.1 GROMACS

GROMACS⁷ is an engine to perform molecular dynamics simulations and energy minimization. These are two of the many techniques that belong to the realm of computational chemistry and molecular modeling [135]. In our experiments, the application ran with 64 MPI tasks and we obtained a trace (application data) comprising just five full iterations.

7.3.2 Computation Bursts and Enhanced Cluster Analysis

To characterize parallel applications structure, the computation bursts are the minimum analysis abstraction used in our analyses. In [73] they define a computation burst, also named CPU burst or simply burst, as the sequential region of the application between communications primitives or calls to a given parallel runtime. This definition is based on the dichotomy that a parallel application can only be performing a parallel primitive or processing data, i.e. computing.

We use the ECII pipeline to group the fine-grain CPU bursts' behavioral trends exhibited along with the application execution. The processor performance hardware counters represent the most interesting piece of information when analyzing the CPU bursts behavior as they provide a unique insight into the CPU performance

⁶<https://www.bsc.es/marenostrum/marenostrum>

⁷<http://www.gromacs.org>

7. Enhanced Cluster Identification and Interpretation Pipeline

at a very fine grain. As a result, we associate each burst with a feature vector of different performance data.

Then we highlight the usefulness of ECII pipeline to ease the characterization of a message-passing parallel application and the ability to detect the finer grain behavior structure in contrast to the existing techniques.

7.3.3 Application Analyses

The analyst/developer has to give thought to lots of different information to analyze a parallel application. For example, Figure 7.4 presents the time-lines of five performances obtained for the GROMACS application executed with 64 tasks. We show the metrics Million of Instructions per Second (MIPS), Instructions per Cycle (IPC), Level I (L1D) data cache misses ratio, and Level II (L2D) data cache misses ratio in timeline 7.4a, 7.4b, 7.4c, and 7.4d respectively since the $X - axis$ is the time, the $Y - axis$ are the application tasks on each time-line, and the color is a gradient from green to blue expressing the magnitude of the given metric.

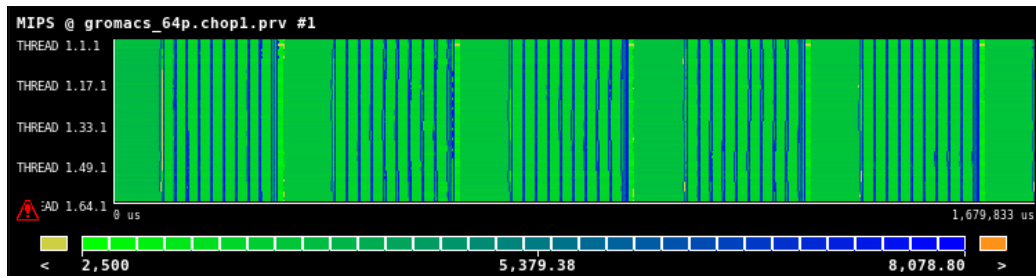
The timelines illustrate a structural pattern includes a series of one wide regular phases followed by nine relatively smaller regions which are followed by small and imbalanced phases (in dark blue on 7.4a and 7.4b). The structural pattern repeats five times. As a result, we could deduce from this pattern that we are observing five iterations of the main loop and nine iteration of the nested loop typically observed in the message-passing applications.

We could use the cluster analysis to draw different conclusions. In [73] they applied DBSCAN to group the CPU bursts characterized. However, they manually selected the appropriate feature subset and tune the DBSCAN algorithm hyper-parameter. They present the subset of hardware counters includes Completed Instructions combined with Instructions per Cycle (IPC) can represent a useful way to determine the application structure.

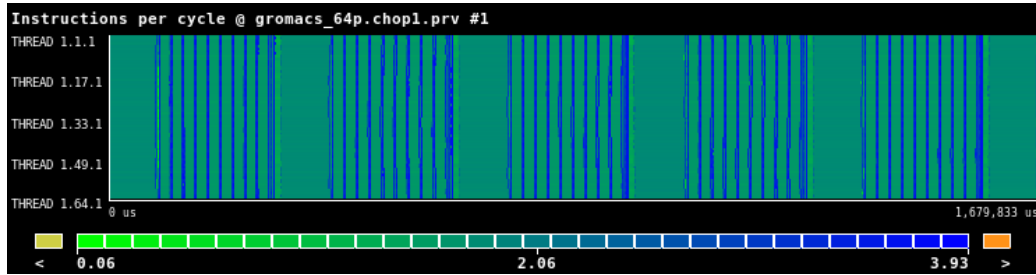
Regarding the DBSCAN hyper-parameter, in [73], they suggest that choosing a quarter of the total tasks the application as the default value of minPts produces the best results in 2-dimensional clustering. Furthermore, they use a histogram with the sorted k-neighbour distance, being k the desired value of minPts. Then this distance is sorted (descending) and plotted. The histogram will show a descending curve. They select as the lower bound of eps, the distance value of the elbow point in the curve, and the distance of the point with the second maximum sorted k-distance as the upper bound of eps. In this example, the minPts equals to 16 (ApplicationTasks/4) and the eps values range is [0.0086, 0.0380] by applying their approach.

Figure 7.5 illustrates the computed search space by the existing approach to tuning the DBSCAN hyper-parameter. Plot 7.5a draws a sorted k-dist graph as

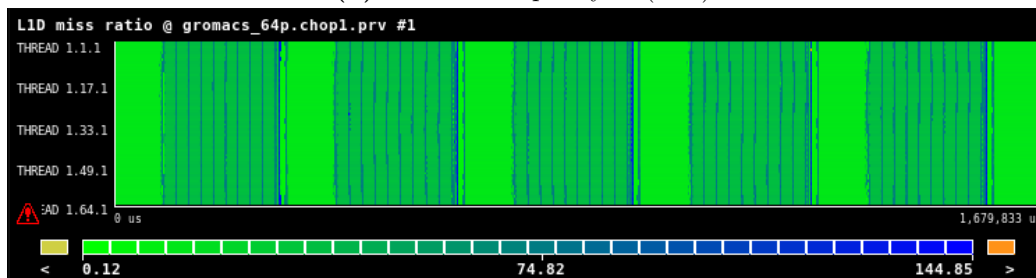
7. Enhanced Cluster Identification and Interpretation Pipeline



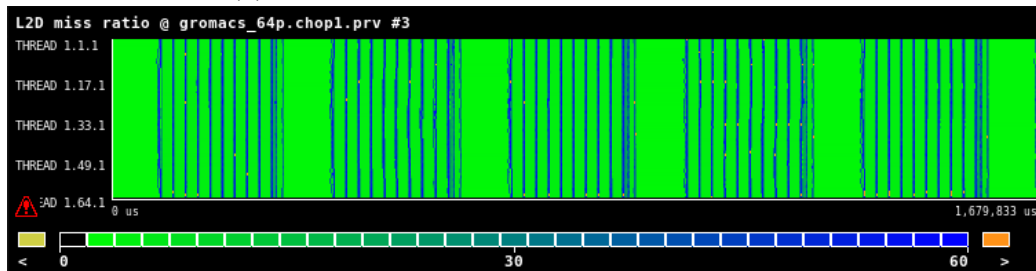
(a) Millions of instructions per second (MIPS)



(b) Instructions per cycle (IPC)



(c) Level I data cache misses per 10^3 instructions.



(d) Level II data cache misses per 10^3 instructions.

Figure 7.4: Time-lines of different performance hardware counter metrics of GROMACS application executed with 64 tasks.

the blue curve, and the two black dotted-lines show the eps lower and upper bound, and the red arrow indicates the eps target interval. Plot 7.5b presents the hyper-parameter search space. Technically, it is a 1D grid search space since it represents a single minPts value. Then, we use an exhaustive grid search to find the optimal eps values through the search space.

Figure 7.6 presents a cluster analysis using a restrictive (small) value of eps (0.0086) and minPts of 16. A small eps value means that the radius of search of the algorithm is shorter, and indeed restrictive, so the results will be a big number of clusters, more

7. Enhanced Cluster Identification and Interpretation Pipeline

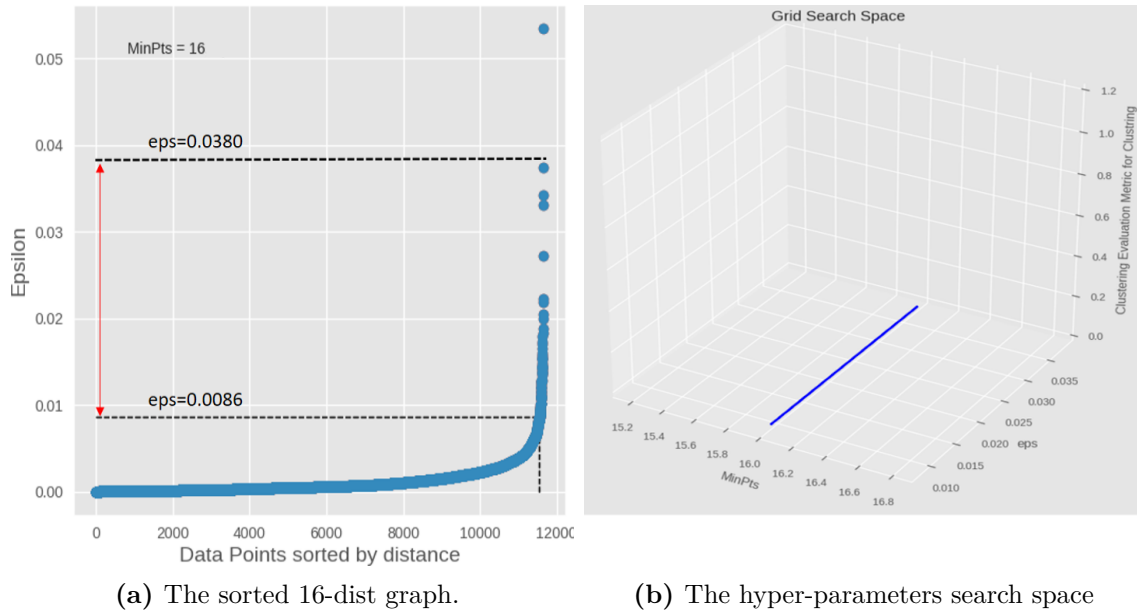


Figure 7.5: The existing approach to tune the DBSCAN hyper-parameters [73]. (a) The sorted 16-dist graph was obtained from the GROMACS application. The blue dots represent the distance to the 16th nearest neighbor for each point in the dataset. We use it to compute the different eps values. (b) The search space to select the optimal eps value.

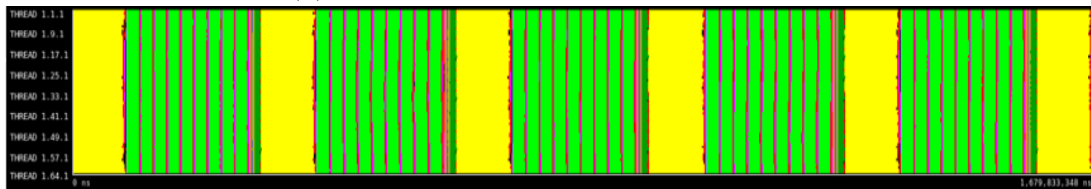
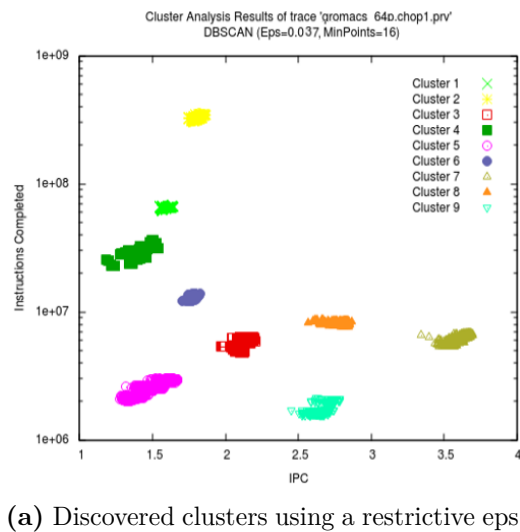
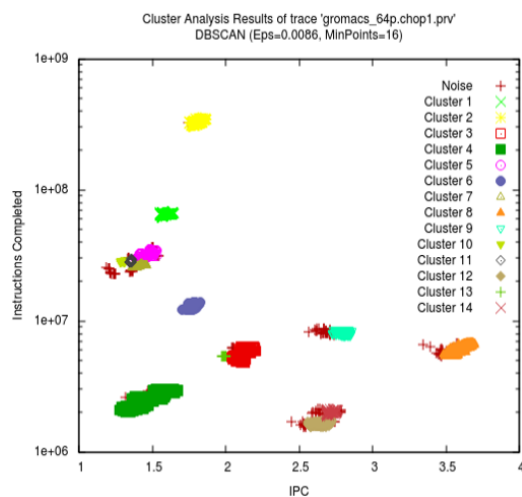


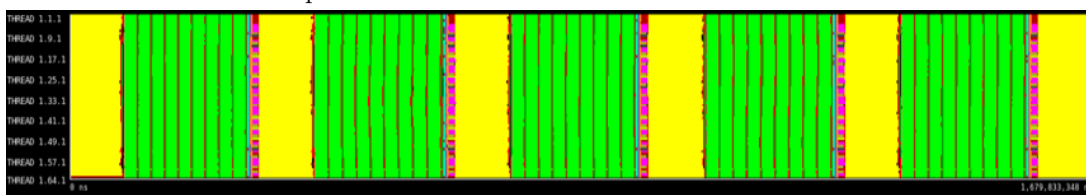
Figure 7.6: Cluster analysis of GROMACS application using DBSCAN clustering algorithm over Completed Instructions and IPC. Due to the use of a restrictive eps (lower bound) value, the detected structure is noisy.

7. Enhanced Cluster Identification and Interpretation Pipeline

compact/dense, and more noise points, as it can be seen in plot 7.6a. In terms of the structure detected, the computation regions each cluster represents will be highly detailed, showing internal behaviors that do not always reflect clear stages in the application execution. As an example, Figure 7.6b shows the timeline containing the clusters detected by DBSCAN. In this Figure, the X – axis of the time-lines is the time and the Y – axis represents the tasks involved in the parallel application execution, and the color indicates the cluster identifier assigned to each CPU burst. The timelines contain five iterations of the application as well. On each iteration we see an initial SPMD phase yellow and successive nine smaller phases green clearly detected while the last part of each iteration does not present the regular SPMD structure we expected.



(a) Discovered clusters using a less restrictive eps.



(b) Distribution of clusters in the time-line

Figure 7.7: A second cluster analysis of GROMACS application using the DBSCAN cluster algorithm over Completed Instructions and IPC. Selecting a higher value of eps by Grid search produces a coarser grain detection, showing an SPMD structure.

On the other hand, the cluster analysis of Figure 7.7 presents the optimal value of the eps which we compute by using an exhaustive grid search through the previously computed search space. Using a selected eps value, 0.037, and the same value for minPts, 16, we obtain a small number of clusters, that aggregate more number of points each, see Figure 7.7a. In this case, the time-line 7.7b shows the detected application structure at a coarser granularity, and the typical SPMD structure appears.

7.3.3.1 Hyper-parameter Tuning

To illustrate the efficiency of our hyper-parameter tuning mechanism, see section 7.2.1, we apply our hyper-parameter tuning approach to improve the clustering result by using the same feature space includes Completed Instructions combined with Instructions per Cycle (IPC).

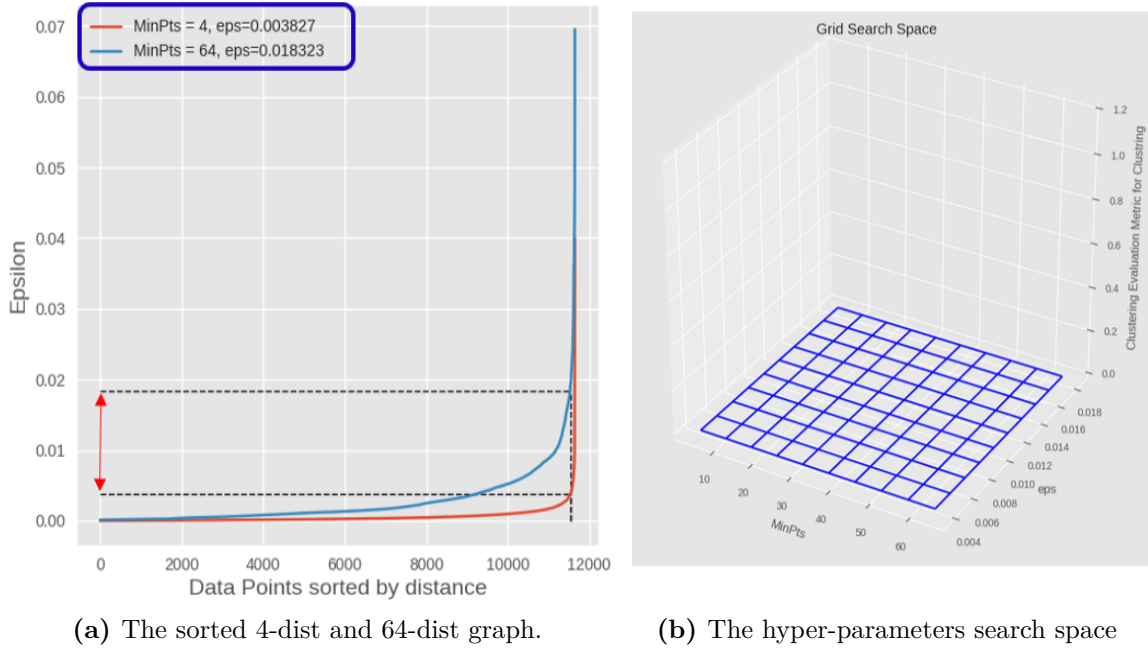


Figure 7.8: Our ECII approach to tune the DBSCAN hyper-parameters over the Completed Instruction and IPC. (a) The sorted 4-dist and 64-dist graph was obtained from the GROMACS application. The blue and red curve represent the distance to the 4th and 64th nearest neighbor for each point in the dataset. We use it to compute the different configuration of the eps and minPts values. (b) The hyper-parameters search space to select the optimal eps value which is a 2D grid.

Figure 7.8 illustrates the computed search space by ECII hyper-parameter tuning mechanism. It computes $[4, 64]$ as minPts targeted search space through Equation 7.4. Plot 7.8a draws a sorted 4-dist and 64-dist graph as the blue and red curves, and the two black dotted-lines show the eps lower and upper bound through Equation 7.5, and the red arrow indicates the eps target interval. Plot 7.8b present the entire hyper-parameter search space. As a result, it is a 2D grid search space.

Then, we use an exhaustive grid search to find the optimal eps and minPts values through the search space. Figure 7.9 shows the result of the grid search. The optimal minPts and eps are 4 and 0.0165 respectively.

In Figure 7.10a we present the results obtained by applying DBSCAN with the parameters minPts = 16 and eps = 0.037 with just tuned eps. We can see in the plot that two clouds inside the black box have been detected as single cluster. To

7. Enhanced Cluster Identification and Interpretation Pipeline

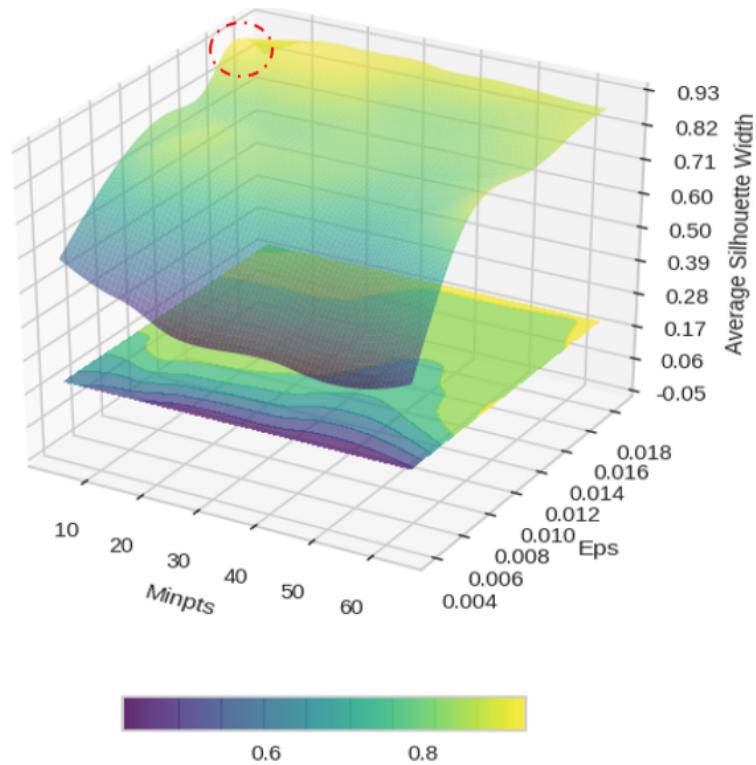


Figure 7.9: The result of our ECII approach to tune the DBSCAN hyper-parameters over the Completed Instruction and IPC. The X - axis, Y - axis, and Z - axis represent the eps, minPts, and Average Silhouette Width values respectively. The red dashed cycle indicates the optimal hyper-parameter configuration.

capture the variability of the cloud inside the box we use the optimal configuration of the both hyper-parameters $\text{minPts} = 4$ and $\text{eps} = 0.0165$. The results of using this minPts and eps are contained in Figure 7.10b. We can see in the scatter plot that this single cloud of the previous analysis now is detected as two distinguished Clusters 9 (light green) and 10 (black). Unfortunately, the optimal parameter configuration provokes a loss-aggregation of the cloud in the blue box. Looking at the time-line 7.10c, we confirm these observations: in the initial section of the fourth iteration, which indicated by number 2 in the time-line, we can see an SPMD region represented by the Cluster 11 (black), while in the other four iterations an almost two times thinner SPMD region represented by the Cluster 10 (light green) at the same position in the sequence, e.g. see number 1 in the time-line.

7.3.3.2 Feature Selection

A common problem when using clustering algorithms is related to the dimensionality of the data. With the performance counters data, we have up to 8 different counters for each CPU burst and several derived metrics with “physical” meaning to the analyst. Throughout this dissertation, we call the counters and derived metrics the

7. Enhanced Cluster Identification and Interpretation Pipeline

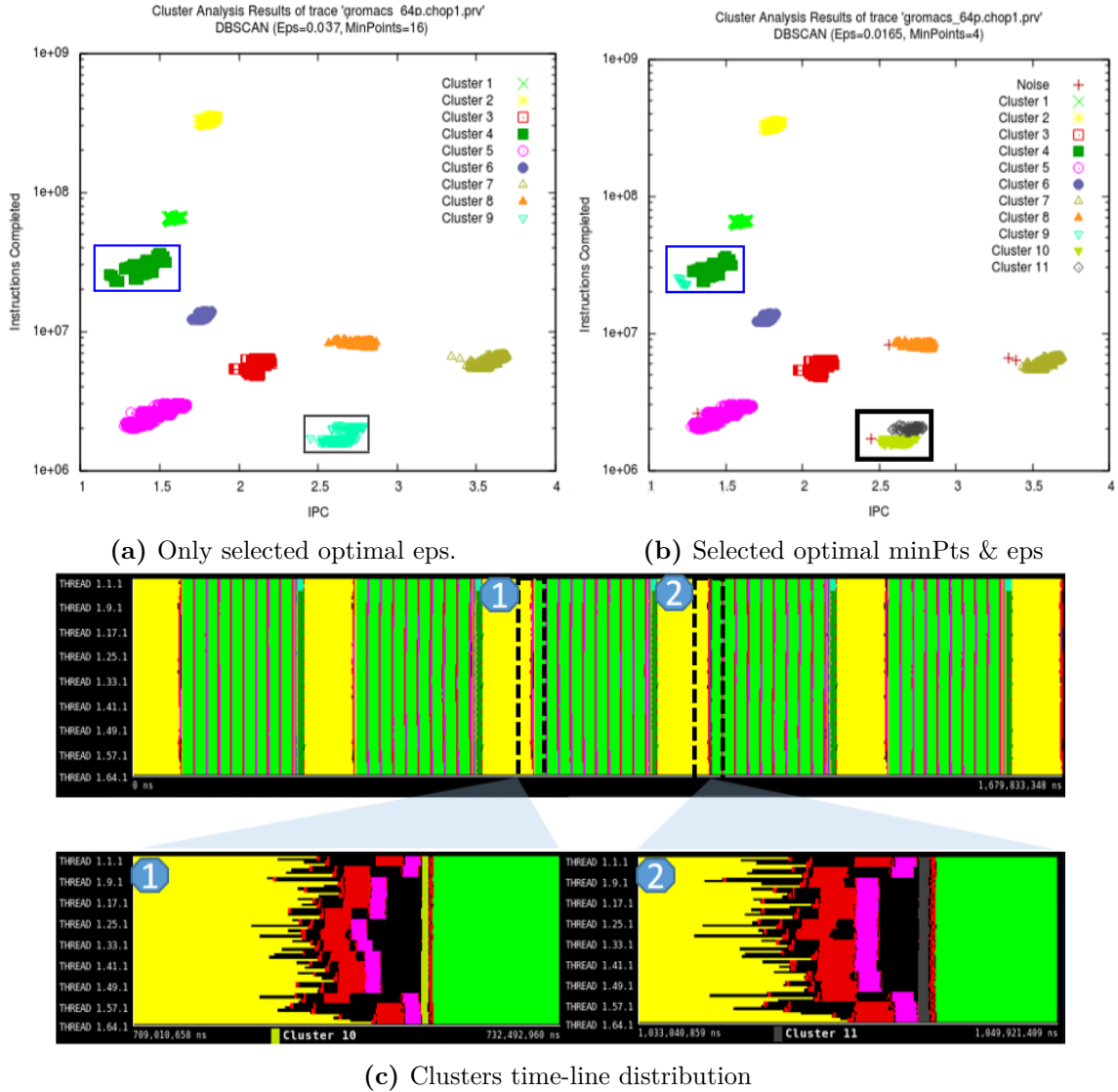


Figure 7.10: Computation structure detection of GROMACS application by applying DBSCAN cluster algorithm with and without the hyper-parameters tuning over the Completed Instruction and IPC.

features. Table 7.1 presents the GROMACS application performance feature set. Our proposal to address this problem is to reduce the dimensionality by selecting features. In our ECII pipeline, we use our RIFS method, see section 4, to select a subset of relevant features for building the DBSCAN model.

We apply our RIFS feature selection approach to improve the clustering results on the GROMACS application by selecting the most two relevant features over the feature set. As a result, it finds the Completed Instruction and L1_Rat as the most relevant feature subset.

In Figure 7.11 we present the results obtained by applying DBSCAN, from the previous section, over Completed Instructions and IPC, and its view over Completed

7. Enhanced Cluster Identification and Interpretation Pipeline

Table 7.1: The GROMACS Application Performance Counters Data.

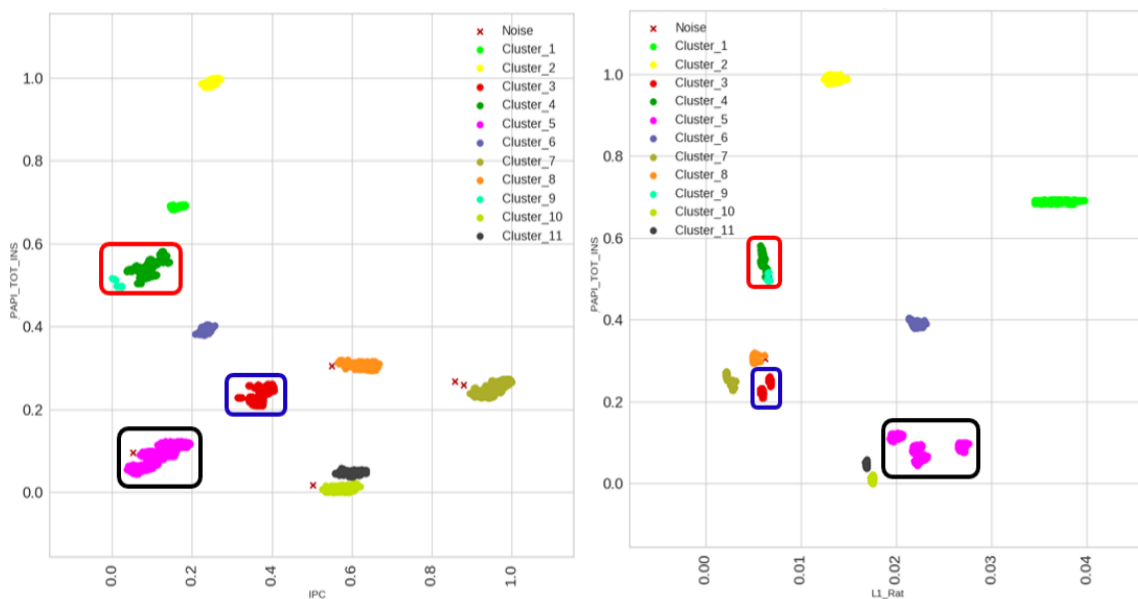
Feature	Description	Type	Category
PAPI_TOT_INS	Completed Instructions	int	Native
PAPI_TOT_CYC	Total cycles	int	Native
PAPI_L1_DCM	L1D cache misses	int	Native
PAPI_L2_DCM	L2D cache misses	int	Native
PAPI_BR_MSP	Branch Misprediction	int	Native
PAPI_BR_INS	Branches	int	Native
R_STALLS:SB	SB full stall cycles	int	Native
R_STALLS:ROB	ROB full stall cycles	int	Native
IPC	PAPI_TOT_INS/PAPI_TOT_CYC	float	Derived
Locality_L1	PAPI_TOT_INS/PAPI_L1_DCM	float	Derived
Locality_L2	PAPI_L1_DCM/PAPI_L2_DCM	float	Derived
L1_Rat	PAPI_L1_DCM/API_TOT_INS	float	Derived
L2_Rat	PAPI_L2_DCM/API_TOT_INS	float	Derived
BRMSP_Rat	PAPI_BR_MSP/API_TOT_INS	float	Derived
BRINS_Rat	PAPI_BR_INS/API_TOT_INS	float	Derived

Instructions and L1_Rat which are selected by applying RIFS algorithm. We can see in the Figure 7.11a that the cloud inside the black and blue box can be detected as four and two different clusters respectively by changing the feature space (view), see Figure 7.11b. On the other hand, the cloud inside the red box contains two Clusters, 4 (dark green) and 9 (light blue), which could be merged as a single cluster. Furthermore, by observing the computation structure these clusters provide in time-line 7.10c, we detect that the clusters contained in the red box reflect pure SPMD regions at the end of each iteration, while it is detected as two separated clusters which is not reflecting the actual SPMD behavior.

Next, we apply the clustering in 2D selected feature space with the tuned hyper-parameters $minPts = 4$ and $eps = 0.0106$. As a result, we obtain the results depicted in Figure 7.12. We can see in the plot how the feature selection step enhances the cluster identification process. It identifies four and two different clusters the clouds present in the black and blue box in 7.11b respectively. It also identifies the cloud inside the red box as a single cluster.

In Figure 7.13 we present the fine-grain computation structure detection of the GROMACS application, using the DBSCAN cluster algorithm over Completed Instructions and L1_Rat. Observing the computation structure of these clusters provide by time-lines 7.13, we detect that the clusters 3 (red) and 5 (purple) presented in the time-line 7.13a reflects SPMD regions at the beginning of each internal repetitive region (light green). However, by looking at the time-line 7.13b, we can see that the two clusters of the previous experiment represent six different SPMD phases

7. Enhanced Cluster Identification and Interpretation Pipeline



(a) Discovered clusters over Completed Instructions and IPC.

(b) A view of the plot (a) over Completed Instructions and L1_Rat.

Figure 7.11: Clustering scatter plot of GROMACS application using clustering algorithm over Completed Instructions and IPC (a), and its view over Completed Instructions and L1_Rat (b). The blue and black boxes highlight clouds of points that can be divided into isolated groups, and the red box highlight the clouds of points that can be detected as an isolated group by changing the feature subset. Note that the coloring is identical.

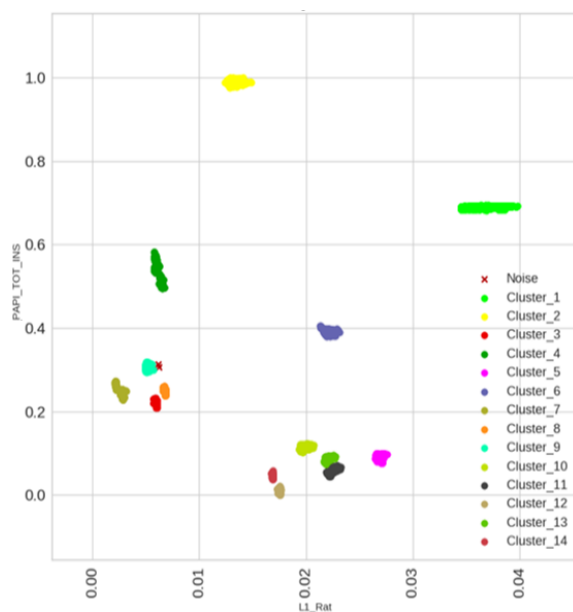
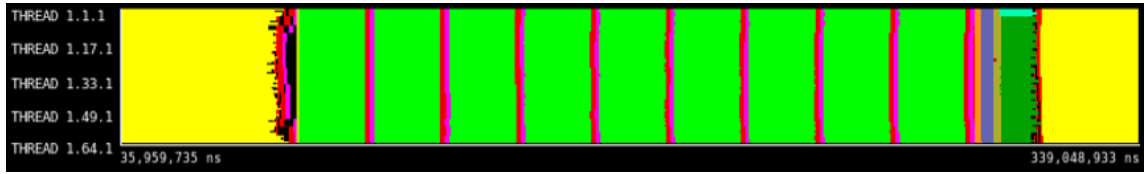


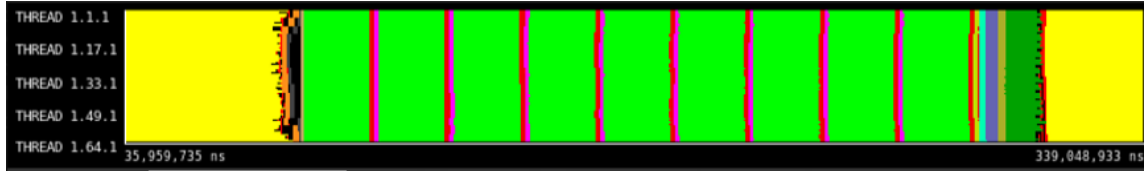
Figure 7.12: Clustering scatter plot of GROMACS application by using the DBSCAN cluster algorithm result over Completed Instructions and L1_Rat with the parameters $\text{minPts} = 4$ and $\text{eps} = 0.0106$.

including Cluster 3 (red), Cluster 8 (orange), Cluster 5 (purple), Cluster 10 (olive green), Cluster 11 (black), and Cluster 13 (green) in the significantly different range

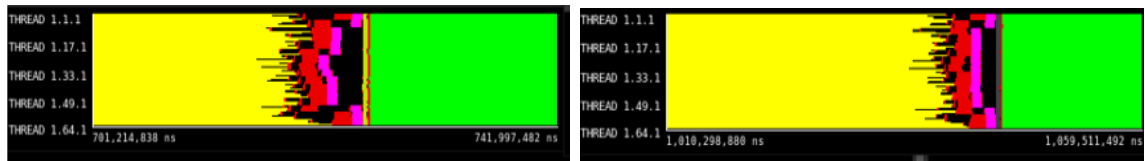
7. Enhanced Cluster Identification and Interpretation Pipeline



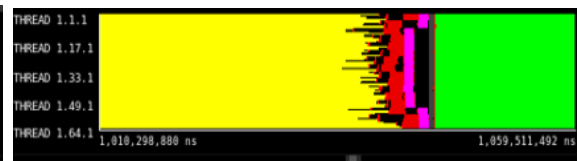
(a) Clusters time-line distribution of the first iteration over Completed Instructions and IPC.



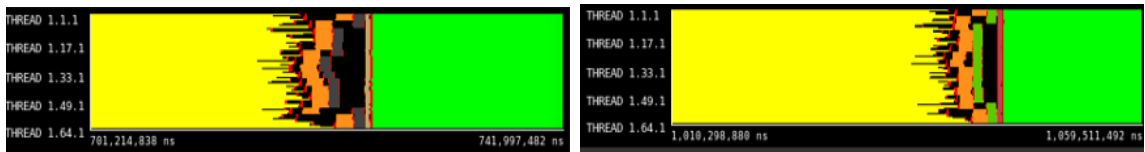
(b) Clusters time-line distribution of the first iteration over Completed Instructions and L1_Rat.



(c) Clusters time-line distribution of the first iteration's initial section over Completed Instructions and IPC.



(d) Clusters time-line distribution of the fourth iteration's initial section over Completed Instructions and IPC.



(e) Clusters time-line distribution of the first iteration's initial section over Completed Instructions and L1_Rat.



(f) Clusters time-line distribution of the fourth iteration's initial section over Completed Instructions and L1_Rat.

Figure 7.13: Computation structure detection of the GROMACS application, using DBSCAN cluster algorithm over Completed Instructions and L1_Rat with the parameters $\text{minPts} = 4$ and $\text{eps} = 0.0106$.

of L1_Rat. Furthermore, by comparing the time-lines 7.13e with 7.13f, we can find Cluster 11 (black) and Cluster 13 (green) also appear in the same range of L1_Rat, but have not been merged. Finally, we see that Cluster 4 (green) detects a clear SPMD phase since it appears in the same range of L1_Rat. We can see that the appropriate feature subset can enhance DBSCAN to select finer grain clusters.

7.3.3.3 Density Homogenization

Afterward, we tackle the failure of DBSCAN to find clusters of varied densities by applying our FSCM algorithm to homogenize the data density.

In Figure 7.14, we compare the DBSCAN clustering quality on original GROMACS performance data and its transformation due to FSCM. It shows that the transformed dataset is stratified more efficiently than the original data by DBSCAN. Plot(a) shows the previous section's DBSCAN clustering result over the Completed Instructions

7. Enhanced Cluster Identification and Interpretation Pipeline

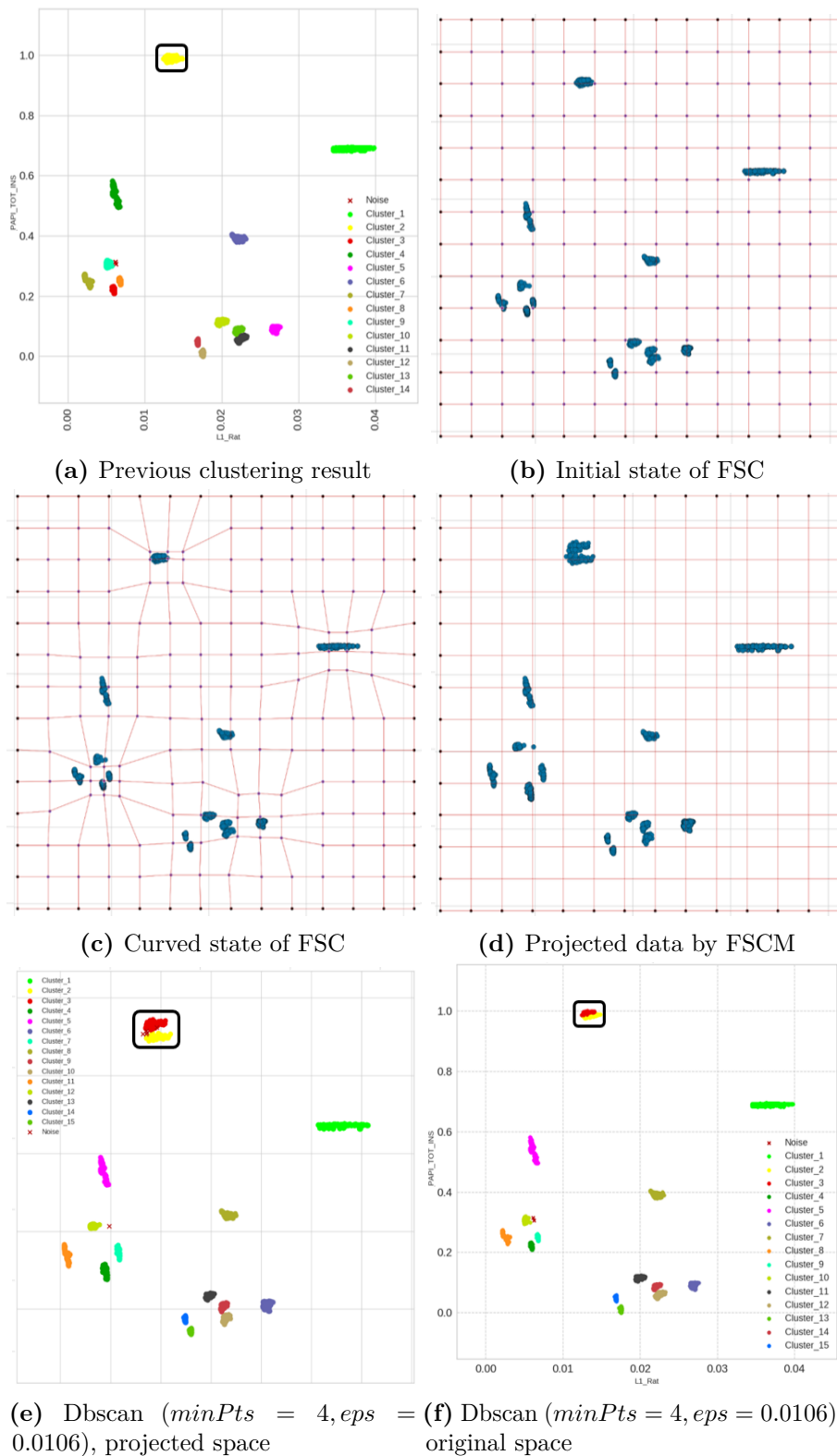
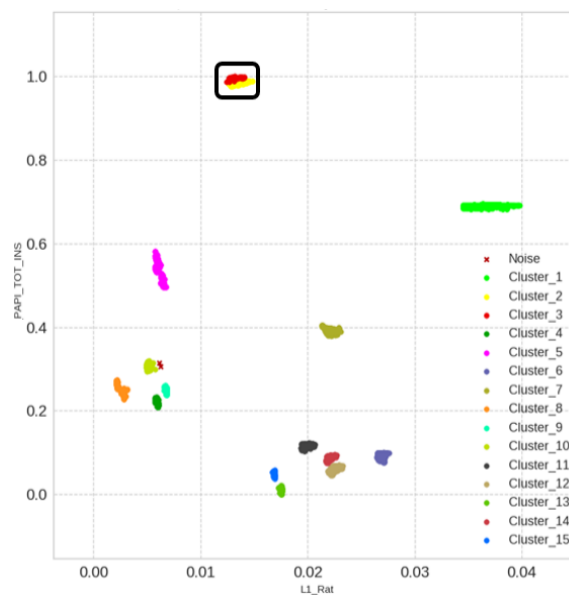


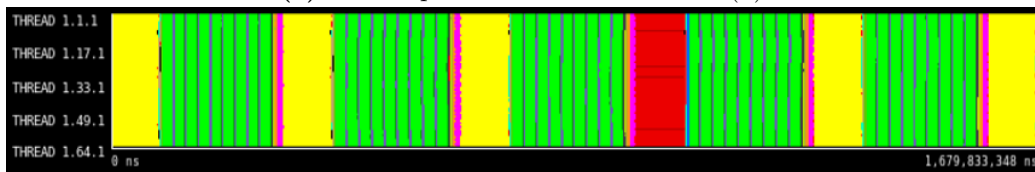
Figure 7.14: Application of the FSCM on the GROMACS application, and using the DBSCAN cluster algorithm over Completed Instructions and L1_Rat on projected new feature space.

7. Enhanced Cluster Identification and Interpretation Pipeline

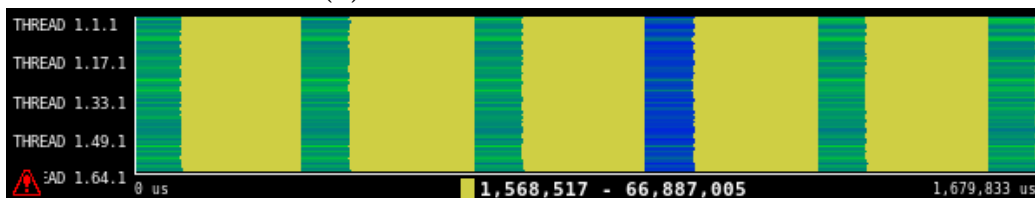
and `L1_Rat`. As shown in plot (a), DBSCAN identified 14 clusters since we observed varied densities in the data. Plots (b) and (c) show the original data scatter plots with the embedded FSC initial state and curved state computed by FSCM respectively. Plot (d) presents the result of the FSCM's bilinear transformation mechanism. As we can see that the cloud of points at the top of the plot gets sparser, while the other data clouds are more separated from each other. Consequently, Cluster 2 (yellow), see the black box in the plot (a), is identified as two clusters Cluster 2 (yellow) and Cluster 3 (red), see the black box in the plot (e). As a result, DBSCAN identified a new distinct cluster by applying a single density threshold, as shown in plot (f).



(a) Scatter-plot of discovered clusters (b)



(b) Clusters time-line distribution



(c) Completed Instruction time-line distribution

Figure 7.15: Computation structure detection of GROMACS application by applying DBSCAN ($minPts = 4, eps = 0.0106$) clustering algorithm over Completed Instructions and `L1_Rat` on the FSCM's homogenized feature space.

In Figure 7.15 we present the finer grain computation structure detection of the GROMACS application by applying the FSCM method, and then using the DBSCAN

7. Enhanced Cluster Identification and Interpretation Pipeline

cluster algorithm over Completed Instructions and L1_Rat. In plot 7.15a, we can see that the Cluster 2 (yellow) of the previous experiment represents two different SPMD phases including Cluster 2 (yellow) and Cluster 3 (red), in the significantly different range of Completed Instruction, see Figure 7.15c. Observing the computation structure of these clusters provided by time-lines 7.15b, we detect that the Cluster 3 (red) presented reflects the SPMD region at the beginning of the fourth internal, while clusters 2 (yellow) represent the initial regions of the other iteration.

7.3.3.4 Extracting Insight

Through this point, we enhanced the cluster analysis by tuning hyper-parameters, using appropriate active features, and homogenizing the density to determine the fine-grain application structure. As a result, we stratified fifteen distinct clusters in different shapes. One can then ask the question, whether each cluster represents a distinct phase of the application why they show heterogeneous performance behavior.

To answer this question, we conduct the OCA method on each cluster to detect the HPC systems bottleneck that can describe the variability among stratified data. In the all conducted experiment, OCA achieves more than 95% topology preservation on average, which shows strong mapping quality.

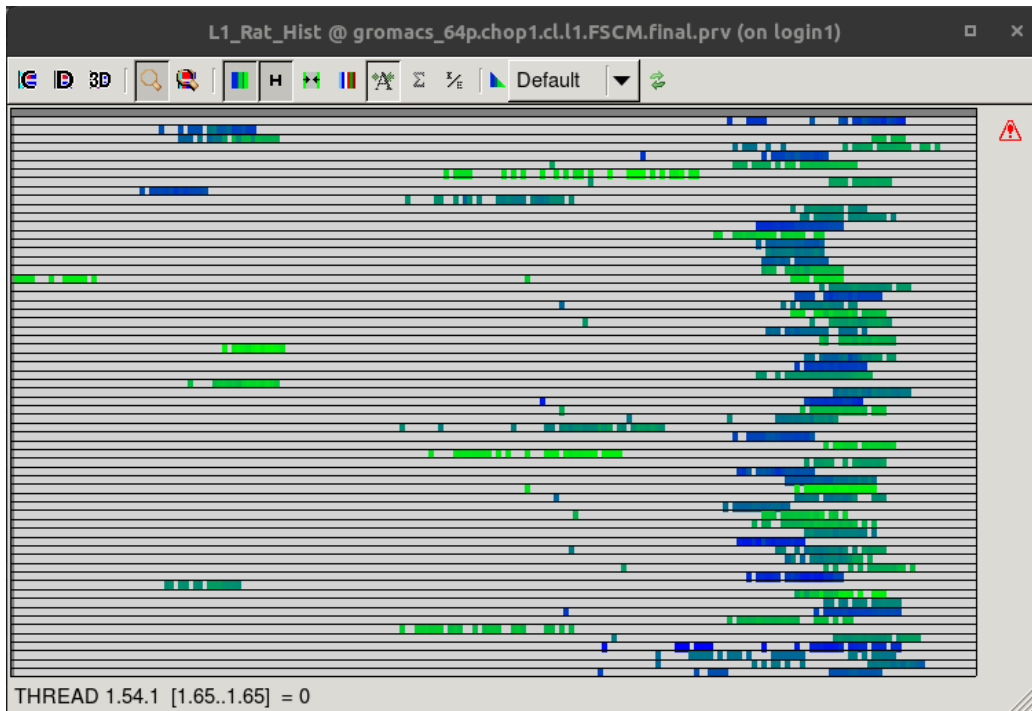
In Table 7.2, we present the computed Directional Sequence Similarity (DSS) for Cluster 1 to Cluster 5. Those clusters approximately represent the 89% of the application execution time. As an example, in case of Cluster 1, the main performance issue could be caused by poor cache usage, as the first organization component exhibits a high DSS value between the Completed Instruction and Locality_L2 (55%). The second organization component exhibits a very high DSS with the Locality_L1 (81%).

Figure 7.16, shows two histograms of the L1_Rat (a) and Locality_L2 (b) of Cluster 1. The $X - axis$ represents duration bins, the $Y - axis$ represents processes, and cells with values indicate that there are computations in the program of that given duration. The color represents the L1_Rat. Locality_L2 for those computations in the plot (a) and plot (b) respectively. As we can see those histograms present significantly similar patterns. As a result, we can conclude that the performance effect of Locality_l1 and Locality_L2 has become the main concern when following the OC1 and OC2 trajectory path. The main performance problem can likely be the L1 and L2 relational capacity. These observations present a useful starting point for the analyst/developer to study what is causing these potential memory problems and improve the regions detected.

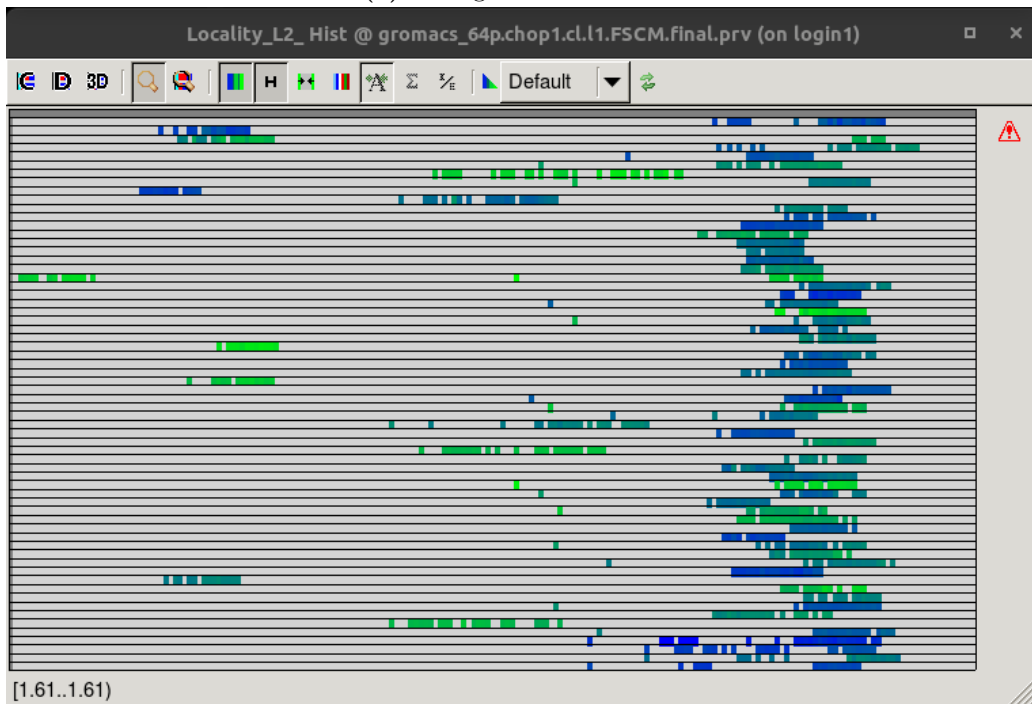
Table 7.2: Application of OCA to GROMACS performance counter data cluster 1 to cluster 5, the feature influencing in the first and second Organization Components.

Feature	Organization Component 1					Organization Component 2				
	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5
IPC	0.40	0.37	0.56	0.11	0.10	0.18	0.53	0.26	0.10	0.11
PAPI_L1_DCM	0.49	0.38	0.44	0.39	0.21	0.32	0.23	0.21	0.37	0.31
PAPI_L2_DCM	0.18	0.33	0.28	0.37	0.36	0.23	0.44	0.13	0.33	0.32
PAPI_BR_MSP	0.20	0.09	0.11	0.47	0.34	0.12	0.14	0.28	0.07	0.05
PAPI_BR_INS	0.07	0.35	0.31	0.29	0.17	0.05	0.40	0.31	0.20	0.17
R_STALLS:SB	0.05	0.18	0.09	0.01	0.09	0.04	0.19	0.03	0.01	0.08
R_STALLS:ROB	0.16	0.24	0.13	0.01	0.11	0.13	0.24	0.06	0.02	0.31
PAPI_TOT_CYC	0.17	0.15	0.19	0.31	0.27	0.11	0.20	0.17	0.33	0.13
Locality_L1	0.42	0.32	0.08	0.15	0.14	0.81	0.07	0.17	0.50	0.20
Locality_L2	0.55	0.51	0.35	0.12	0.15	0.46	0.26	0.42	0.10	0.34
L2_Rat	0.14	0.26	0.22	0.14	0.20	0.10	0.23	0.20	0.12	0.46
BRMSP_Rat	0.12	0.08	0.09	0.40	0.18	0.12	0.09	0.36	0.05	0.06
BRINS_Rat	0.14	0.10	0.41	0.06	0.43	0.11	0.12	0.31	0.05	0.15

7. Enhanced Cluster Identification and Interpretation Pipeline



(a) Histograms of L1_Rat.



(b) Histograms of Locality_L2.

Figure 7.16: Histograms of Locality_L2 and L1_Rat for Cluster 1 in GROMACS application. The patterns of both metrics are almost identical.

8

Conclusion

Contents

8.1	Feature Selection for Noisy Data	149
8.2	Extracting Insights from the Shape of Cluster	150
8.3	Homogenizing the Clusters Density	150
8.4	Enhanced Cluster Identification and Interpretation Pipeline	151
8.5	Future Work	152

Previous chapters of this thesis have presented and demonstrated the usefulness of the main contribution of this thesis: the ECII clustering pipeline and its novel components. We presented the research we conducted using a machine learning pipeline to enhance the cluster analysis producer, and particularly in the parallel performance analysis scenario. As contributions to the machine learning area, we proposed and validated new feature selection technique for the noisy data, a multi-linear transformation to homogenize the cluster density, and a novel approach to extract insight from the clustering result. We also demonstrated the utility of the techniques introduced by using them to perform clustering analyses of parallel applications.

8.1 Feature Selection for Noisy Data

This thesis has presented a new robust unsupervised feature selection approach called, *Robust Independent Feature Selection* (RIFS). We propose to make the best use of the independent components structure of a set of features, which is defined on the mixing matrix, both to select the feature subset and to decouple the noise as a latent independent source, simultaneously. Thus, RIFS isolates the noise by rotating

8. Conclusion

the mixing matrix obliquely. When we have compared our RIFS method with two state-of-the-art methods, namely, Laplacian Score and Max Variance, the empirical results on different real world data sets validate that the proposed method obtains considerably higher effectiveness for both clustering and classification. Our proposed RIFS algorithm performs well on original data and it is strongly resists noises.

8.2 Extracting Insights from the Shape of Cluster

We have presented a new topology-preserving approach to study the complex and arbitrary shape of the stratified data called *Organization Component Analysis* (OCA). We propose to make the best use of the self-organizing map structure of a high dimensional categorized data, which is defined on the 2D grid of neurons, both to recognize and quantify innate cluster structure and its formation, simultaneously. Whereas cluster analysis identifies regions of higher density in these data, OCA is able to extract finer-grain insights from the shape of a cluster, as it is clearly demonstrated in this article. Here OCA is a general and an efficient method that is assumption free, automated, and it can be applied on the result of any clustering algorithm. Moreover, OCA creates a graph to visualize the shape of these clusters by way of a graph. Furthermore, we propose a novel Directional Sequence Similarity method to compute the similarity between two sequences of changes, in which the rate of change is taken along a unit vector. Finally, we have shown that our novel topology-preserving approach can lead to finer and profounder insights of two real-world datasets. The usefulness of our OCA technique is not closed to these two types of applications but can generally be applied to diverse data types, such as time series, image segmentation, consumer behavior data and others.

8.3 Homogenizing the Clusters Density

Furthermore, we have presented a new topological Feature Space Curvature Map (FSCM) method to homogenize the density of data to overcome the weakness of density-based clustering algorithms in finding clusters of varied densities. Our FSCM involves two steps: Feature Space Curvature modeling and Curvature Mapping to non-linearly project a multi-dimensional dataset. In analogy to Relativity Theory, we assume an m -dimensional feature space as an elastic Feature Space Fabric (FSF) which is bent when data points are placed in it depending on their density. Therefore, the massive data clouds wrap the FSC intensively while the sparse ones wrap it slightly. We propose a new gravitation-based version of self-organizing map (GSOM) to model the density structure of the data, which is defined on the m -dimensional

8. Conclusion

Regular Rectangular Grid (RRG) of neurons to recognize data density structure. In consequence, GSOM guarantees the smoothness and continuity of the final FSC model and it converges faster to a stable model than standard SOM. Then, we apply a novel multilinear transformation to straighten out the wrapped FSC to reach a new equidistant feature space when the data points are attached to the Feature Space Fabric. Our parametric multilinear transformation approach projects the data points to the new feature space, showing more uniform density among data aggregations than in the original space. As a result, existing density-based clustering algorithms can efficiently identify clusters within this data homogenized by FSCM, as it is clearly demonstrated empirically in this thesis. Here FSCM is a general and an efficient preprocessing method that is parametric, assumption free and automated that can be applied before clustering analysis. This often leads to better not only clustering analysis results than using original data with the advantages of being able to homogenize the density.

Finally, we have shown that our novel FSCM approach can efficiently improve the clustering performance of three state-of-art density-based algorithms, DBSCAN, OPTICS and DP with regard to *FScore* in nearly all datasets we have used in the experiments. The usefulness of our FSCM technique is not closed to these algorithms and datasets but can generally be applied to diverse combination of clustering methods and dataset. In addition, FSCM surpasses both competitor CDF transform methods ReScale and DScale since the prior just scales one-dimensional and the latter is not included point-shifting and it doesn't product a measurable metric output. Moreover, FSCM is more generalized than these previously aforementioned transform methods since it is a assumption free model-based approach permits both potentially complex multi-density clusters and various cluster shapes. Furthermore, FSCM enables an existing density-based algorithm to identify clusters of various densities in a less computational cost than both ReScale and DScale methods and has less parameter tuning than ReScale.

8.4 Enhanced Cluster Identification and Interpretation Pipeline

After acquiring some expertise with the clustering analysis technique based on the DBSCAN algorithm we detected the limitations of this algorithm. First, the DBSCAN parameters could be a handicap for a non-expert user, and second, and more importantly, for those inputs with different densities across the data space, the use of a single hyper-parameter configuration limits the ability to correctly detect the actual clusters where the dataset presents multiple densities. We proposed the Enhanced Cluster Identification and Interpretation (ECII) Pipeline to overcome these problems.

8. Conclusion

ECII pipeline iteratively improves the quality of the generic DBSCAN algorithm, using the Average Silhouette Width (ASW) Criterion to evaluate the resulting clusters in each iteration. The proposed algorithm also takes advantage of the common points of DBSCAN and the feature selection method. These methods basically select the most relevant feature subset and then use the FSCM algorithm to homogenize the density of the data as a preprocessing step. In DBSCAN, the different higher and lower eps values applied to correspond to the higher and lower value of the minPts is used as search space.

Then it generates a set of minPts and generates N increasingly sorted eps values, and it applies an exhaustive grid search to look for the best hyper-parameter configuration automatically. On each iteration it applies DBSCAN using minPts and the corresponding eps to the input data set, evaluating the resulting clusters using a quality score. Those individuals that belong to clusters that pass the score are discarded in further iterations. The algorithm finishes after N iterations.

Using the ECII with the CPU burst data of a parallel application, and applying the Average Silhouette Width to evaluate the clusters, we provide the developer/analyst a high-quality SPMD computation structure detection with the added value that reflects the fine grain of the computation regions. In addition, the algorithm just requires the data set as input, but no other parameters.

8.5 Future Work

The improvement of the density-based clustering algorithms, particularly to make them scalable to large and potentially complex datasets, has always been an interesting topic for the clustering analysis community. The work presented here can be seen as a first step toward developing an automated cluster analysis pipeline. There are many ways in which this pipeline can be improved, modified for different applications, or extended to other domains by embedding more data preprocessing steps and using different clustering algorithms.

From the algorithmic point of view, we are working on improving the performance of GSOM by relaxing the rigid boundary condition. We think the boundary nodes weight vector can be updated aligned with their axes. As a result, the GSOM can extract the finer-grain density structure of the data. We have also provided a small study to give an insight into the effect of various distance metrics on the methods used in the study.

Furthermore, we would like to come up with a hybrid clustering approach to obtain very detailed structure identification by giving the outer level flexibility to operate on an approximate coarse grain euclidean space with DBSCAN and great detail on non-homogeneous local space deformations with GSOM.

8. Conclusion

From the practical point of view, the proposed version of ECII pipeline has been applied to real-life applications in several domains such as performance, machinery, biological data, etc. We are looking for new applications and domains to apply our purposed methods.

Bibliography

- [1] James Brandt et al. “Quantifying effectiveness of failure prediction and response in HPC systems: Methodology and example”. In: *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE. 2010, pp. 2–7.
- [2] *Cisco bug: Csctf52095 - manually flushing os cache during load impacts server*. <https://quickview.cloudapps.cisco.com/quickview/bug/CSCtf52095>. Accessed: 2017-02-07.
- [3] Anthony Agelastos et al. “Toward rapid understanding of production HPC applications and systems”. In: *2015 IEEE International Conference on Cluster Computing*. IEEE. 2015, pp. 464–473.
- [4] James M Brandt et al. *Enabling Advanced Operational Analysis Through Multi-subsystem Data Integration on Trinity*. Tech. rep. Sandia National Lab.(SNL-CA), Livermore, CA (United States); Sandia National . . . , 2015.
- [5] Abhinav Bhatele et al. “There goes the neighborhood: performance degradation due to nearby jobs”. In: *SC’13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE. 2013, pp. 1–12.
- [6] Matthieu Dorier et al. “CALCioM: Mitigating I/O interference in HPC systems through cross-application coordination”. In: *2014 IEEE 28th international parallel and distributed processing symposium*. IEEE. 2014, pp. 155–164.
- [7] Anthony Agelastos et al. “The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications”. In: *SC’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2014, pp. 154–165.
- [8] Juan Gonzalez, Judit Gimenez, and Jesus Labarta. “Automatic detection of parallel applications computation phases”. In: *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE. 2009, pp. 1–11.
- [9] Janos Abonyi and Balázs Feil. *Cluster analysis for data mining and system identification*. Springer Science & Business Media, 2007.
- [10] Martin Ester et al. “Density-based spatial clustering of applications with noise”. In: *Int. Conf. Knowledge Discovery and Data Mining*. Vol. 240. 1996, p. 6.
- [11] Mihael Ankerst et al. “OPTICS: Ordering points to identify the clustering structure”. In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.
- [12] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).
- [13] Fatima Batool and Christian Hennig. “Clustering with the average silhouette width”. In: *Computational Statistics & Data Analysis* 158 (2021), p. 107190.

Bibliography

- [14] Kaveh Mahdavi, Jesus Labarta, and Judit Gimenez. “Unsupervised Feature Selection for Noisy Data”. In: *International Conference on Advanced Data Mining and Applications*. Springer. 2019, pp. 79–94.
- [15] Kaveh Mahdavi, Jesus Labarta Mancho, and Judit Gimenez Lucas. “Organization Component Analysis: The method for extracting insights from the shape of cluster”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–10.
- [16] Kaveh Mahdavi, Jesus Labarta Mancho, and Judit Gimenez Lucas. “Feature Space Curvature Map: A Method To Homogenize The Density”. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2022, pp. 1–10.
- [17] Allen D Malony et al. “Parallel performance measurement of heterogeneous parallel systems with gpus”. In: *2011 international conference on parallel processing*. IEEE. 2011, pp. 176–185.
- [18] Shirley Browne et al. “A portable programming interface for performance evaluation on modern processors”. In: *The international journal of high performance computing applications* 14.3 (2000), pp. 189–204.
- [19] Dan Terpstra et al. “Collecting performance data with PAPI-C”. In: *Tools for High Performance Computing 2009*. Springer, 2010, pp. 157–173.
- [20] Allen D Malony et al. “An experimental approach to performance measurement of heterogeneous parallel applications using cuda”. In: *Proceedings of the 24th ACM international Conference on Supercomputing*. 2010, pp. 127–136.
- [21] Ewing Lusk et al. “MPI: A message-passing interface standard”. In: *International Journal of Supercomputer Applications* 8.3/4 (2009), p. 623.
- [22] Susan L Graham, Peter B Kessler, and Marshall K McKusick. “Gprof: A call graph execution profiler”. In: *ACM Sigplan Notices* 39.4 (2004), pp. 49–57.
- [23] M Schulz et al. “Analyzing the Performance of Scientific Applications with Open|SpeedShop”. In: *Parallel Computational Fluid Dynamics: Recent Advances and Future Directions* (2010), p. 151.
- [24] Laksono Adhianto et al. “HPCToolkit: Tools for performance analysis of optimized parallel programs”. In: *Concurrency and Computation: Practice and Experience* 22.6 (2010), pp. 685–701.
- [25] Sameer S Shende and Allen D Malony. “The TAU parallel performance system”. In: *The International Journal of High Performance Computing Applications* 20.2 (2006), pp. 287–311.
- [26] Markus Geimer et al. “The Scalasca performance toolset architecture”. In: *Concurrency and computation: Practice and experience* 22.6 (2010), pp. 702–719.
- [27] Zoltán Szebenyi, Felix Wolf, and Brian JN Wylie. “Space-efficient time-series call-path profiling of parallel applications”. In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. 2009, pp. 1–12.
- [28] Barcelona Supercomputing Center. “CEPBA-Tools. Paraver. Parallel Program Visualization and Analysis tool Version 3.0”. In: vol. 1. 1. Citeseer. 2001.
- [29] *Score-P Project Homepage*. <http://www.score-p.org>.

Bibliography

- [30] Holger Brunst and Bernd Mohr. “Performance analysis of large-scale OpenMP and hybrid MPI/OpenMP applications with Vampir NG”. In: *International Workshop on OpenMP*. Springer. 2005, pp. 5–14.
- [31] Dominic Eschweiler et al. “Open trace format 2: The next generation of scalable trace formats and support libraries”. In: *Applications, Tools and Techniques on the Road to Exascale Computing*. IOS Press, 2012, pp. 481–490.
- [32] Andreas Knüpfer et al. “Introducing the open trace format (OTF)”. In: *International Conference on Computational Science*. Springer. 2006, pp. 526–533.
- [33] Michael Wagner, Andreas Knupfer, and Wolfgang E Nagel. “Enhanced encoding techniques for the open trace format 2”. In: *Procedia Computer Science* 9 (2012), pp. 1979–1987.
- [34] Marty Itzkowitz et al. “Memory profiling using hardware counters”. In: *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. 2003, p. 17.
- [35] Nathan R Tallent et al. “Scalable fine-grained call path tracing”. In: *Proceedings of the international conference on Supercomputing*. 2011, pp. 63–74.
- [36] Harald Servat et al. “Detailed performance analysis using coarse grain sampling”. In: *European Conference on Parallel Processing*. Springer. 2014, pp. 185–198.
- [37] Gary Lakner et al. “IBM System Blue Gene Solution: Performance Analysis Tools”. In: *IBM Redpaper Publication* (2008).
- [38] Laksono Adhianto, John Mellor-Crummey, and Nathan R Tallent. “Effectively presenting call path profiles of application performance”. In: *2010 39th International Conference on Parallel Processing Workshops*. IEEE. 2010, pp. 179–188.
- [39] John Mellor-Crummey et al. “HPCView: A tool for top-down analysis of node performance”. In: *The Journal of Supercomputing* 23.1 (2002), pp. 81–104.
- [40] John Mellor-Crummey et al. *HPCToolkit User’s Manual*. 2012.
- [41] Markus Geimer et al. “Scalable Collation and Presentation of Call-Path Profile Data with CUBE.” In: *PARCO*. 2007, pp. 645–652.
- [42] Robert Bell, Allen D Malony, and Sameer Shende. “Paraprof: A portable, extensible, and scalable tool for parallel performance profile analysis”. In: *European Conference on Parallel Processing*. Springer. 2003, pp. 17–26.
- [43] Kevin A Huck et al. “Design and implementation of a parallel performance data management framework”. In: *2005 International Conference on Parallel Processing (ICPP’05)*. IEEE. 2005, pp. 473–482.
- [44] J Mark Bull. “A hierarchical classification of overheads in parallel programs”. In: *Software Engineering for Parallel and Distributed Systems*. Springer, 1996, pp. 208–219.
- [45] Felix Wolf and Bernd Mohr. “Automatic performance analysis of MPI applications based on event traces”. In: *European Conference on Parallel Processing*. Springer. 2000, pp. 123–132.
- [46] Felix Wolf and Bernd Mohr. “Automatic performance analysis of hybrid MPI/OpenMP applications”. In: *Journal of Systems Architecture* 49.10-11 (2003), pp. 421–439.

Bibliography

- [47] Felix Wolf et al. “Efficient pattern search in large traces through successive refinement”. In: *European Conference on Parallel Processing*. Springer. 2004, pp. 47–54.
- [48] Markus Geimer et al. “Scalable parallel trace-based performance analysis”. In: *European Parallel Virtual Machine/Message Passing Interface Users’ Group Meeting*. Springer. 2006, pp. 303–312.
- [49] David Böhme et al. “Identifying the root causes of wait states in large-scale parallel applications”. In: *ACM Transactions on Parallel Computing (TOPC)* 3.2 (2016), pp. 1–24.
- [50] Antonio Espinosa, Tomas Margalef, and Emilio Luque. “Automatic performance evaluation of parallel programs”. In: *Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing-PDP’98-*. IEEE. 1998, pp. 43–49.
- [51] Josep Jorba, Tomàs Margalef, and Emilio Luque. “Performance Analysis of Parallel Applications with KappaPI 2.” In: *PARCO*. Citeseer. 2005, pp. 155–162.
- [52] Thomas Fahringer, Michael Gerndt, Graham Riley, et al. “Knowledge specification for automatic performance analysis”. In: (1999).
- [53] Hong-Linh Truong and Thomas Fahringer. “SCALEA: A performance analysis tool for distributed and parallel programs”. In: *European Conference on Parallel Processing*. Springer. 2002, pp. 75–85.
- [54] Clovis Seragiotto et al. “On using aksum for semi-automatically searching of performance problems in parallel and distributed programs”. In: *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2003. Proceedings*. IEEE. 2003, pp. 385–392.
- [55] Thomas Fahringer and Clovis Seragiotto. “Aksum: A performance analysis tool for parallel and distributed applications”. In: *Performance analysis and grid computing*. Springer, 2004, pp. 189–208.
- [56] Barton P Miller et al. “The Paradyn parallel performance measurement tool”. In: *Computer* 28.11 (1995), pp. 37–46.
- [57] Philip C Roth and Barton P Miller. “On-line automated performance diagnosis on thousands of processes”. In: *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*. 2006, pp. 69–80.
- [58] Michael Gerndt et al. “Performance analysis for teraflop computers: a distributed automatic approach”. In: *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*. IEEE. 2002, pp. 23–30.
- [59] Michael Gerndt, Karl Furlinger, and Edmond Kereku. “Periscope: Advanced Techniques for Performance Analysis.” In: *ParCo*. 2005, pp. 15–26.
- [60] Jeffrey K Hollingsworth, Barton Paul Miller, and Jon Cargille. “Dynamic program instrumentation for scalable performance tools”. In: *Proceedings of IEEE Scalable High Performance Computing Conference*. IEEE. 1994, pp. 841–850.
- [61] Bryan Buck and Jeffrey K Hollingsworth. “An API for runtime code patching”. In: *The International Journal of High Performance Computing Applications* 14.4 (2000), pp. 317–329.
- [62] Marc Casas, Rosa M Badia, and Jesus Labarta. “Automatic structure extraction from MPI applications tracefiles”. In: *European Conference on Parallel Processing*. Springer. 2007, pp. 3–12.

Bibliography

- [63] Felix Freitag, Julita Corbalan, and Jesus Labarta. “A dynamic periodicity detector: Application to speedup computation”. In: *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*. IEEE. 2001, 6–pp.
- [64] Andreas Knüpfer et al. “Visualization of repetitive patterns in event traces”. In: *International Workshop on Applied Parallel Computing*. Springer. 2006, pp. 430–439.
- [65] Andreas Knüpfer and Wolfgang E Nagel. “New algorithms for performance trace analysis based on compressed complete call graphs”. In: *International Conference on Computational Science*. Springer. 2005, pp. 116–123.
- [66] Oleg Y Nickolayev, Philip C Roth, and Daniel A Reed. “Real-time statistical clustering for event trace reduction”. In: *The International Journal of Supercomputer Applications and High Performance Computing* 11.2 (1997), pp. 144–159.
- [67] Philip Charles Roth et al. “ETRUSCA, event trace reduction using statistical data clustering analysis”. MA thesis. Citeseer, 1996.
- [68] Dong H Ahn and Jeffrey S Vetter. “Scalable analysis techniques for microprocessor performance counter metrics”. In: *SC’02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. IEEE. 2002, pp. 3–3.
- [69] Rasmus Bro and Age K Smilde. “Principal component analysis”. In: *Analytical methods* 6.9 (2014), pp. 2812–2831.
- [70] Paul Kline. *An easy guide to factor analysis*. Routledge, 2014.
- [71] Kevin A Huck et al. “Knowledge support and automation for performance analysis with PerfExplorer 2.0”. In: *Scientific programming* 16.2-3 (2008), pp. 123–134.
- [72] Timothy Sherwood et al. “Automatically characterizing large scale program behavior”. In: *ACM SIGPLAN Notices* 37.10 (2002), pp. 45–57.
- [73] Juan Gonzalez et al. “Automatic refinement of parallel applications structure detection”. In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE. 2012, pp. 1680–1687.
- [74] Pavel Berkhin. “A survey of clustering data mining techniques”. In: *Grouping multidimensional data*. Springer, 2006, pp. 25–71.
- [75] Dongkuan Xu and Yingjie Tian. “A comprehensive survey of clustering algorithms”. In: *Annals of Data Science* 2.2 (2015), pp. 165–193.
- [76] Jyoti Yadav and Monika Sharma. “A Review of K-mean Algorithm”. In: *Int. J. Eng. Trends Technol* 4.7 (2013), pp. 2972–2976.
- [77] Tsunenori Ishioka et al. “An expansion of X-means for automatically determining the optimal number of clusters”. In: *Proceedings of International Conference on Computational Intelligence*. Vol. 2. Citeseer. 2005, pp. 91–95.
- [78] Hae-Sang Park and Chi-Hyuck Jun. “A simple and fast algorithm for K-medoids clustering”. In: *Expert systems with applications* 36.2 (2009), pp. 3336–3341.
- [79] Frank Nielsen. “Hierarchical clustering”. In: *Introduction to HPC with MPI for Data Science*. Springer, 2016, pp. 195–211.
- [80] Martin Ester et al. “Density-based spatial clustering of applications with noise”. In: 240.6 (1996).

Bibliography

- [81] Nadia Rahmah and Imas Sukaesih Sitanggang. “Determination of optimal epsilon (eps) value on dbSCAN algorithm to clustering data on peatland hotspots in sumatra”. In: *IOP conference series: earth and environmental science*. Vol. 31. 1. IOP Publishing. 2016, p. 012012.
- [82] *Estimate neighborhood clustering threshold - MATLAB clusterDBSCAN.estimateEpsilon MathWorks Spain*. <https://es.mathworks.com/help/radar/ref/clusterdbscan.clusterdbscan.estimateepsilon.html>. (Accessed on 04/01/2022).
- [83] Ville Satopaa et al. “Finding a "knee" in a haystack: Detecting knee points in system behavior”. In: *2011 31st international conference on distributed computing systems workshops*. IEEE. 2011, pp. 166–171.
- [84] Jieli Wang et al. “Unsupervised learning of topological phase transitions using the Calinski-Harabaz index”. In: *Physical Review Research* 3.1 (2021), p. 013074.
- [85] Slobodan Petrovic. “A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters”. In: *Proceedings of the 11th Nordic workshop of secure IT systems*. Vol. 2006. Citeseer. 2006, pp. 53–64.
- [86] Artur Starczewski and Adam Krzyżak. “Performance evaluation of the silhouette index”. In: *International conference on artificial intelligence and soft computing*. Springer. 2015, pp. 49–58.
- [87] James C Bezdek and Nikhil R Pal. “Some new indexes of cluster validity”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.3 (1998), pp. 301–315.
- [88] Olatz Arbelaiz et al. “An extensive comparative study of cluster validity indices”. In: *Pattern recognition* 46.1 (2013), pp. 243–256.
- [89] Katherine S Pollard and Mark J Van Der Laan. “A method to identify significant clusters in gene expression data”. In: (2002).
- [90] Steffen Huber et al. “DMME: Data mining methodology for engineering applications—a holistic extension to the CRISP-DM model”. In: *Procedia Cirp* 79 (2019), pp. 403–408.
- [91] Sayan Putatunda. *Practical Machine Learning for Streaming Data with Python*. Springer, 2021.
- [92] Petro Liashchynskyi and Pavlo Liashchynskyi. “Grid search, random search, genetic algorithm: A big comparison for NAS”. In: *arXiv preprint arXiv:1912.06059* (2019).
- [93] Peter I Frazier. “A tutorial on Bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).
- [94] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.
- [95] Xinjie Fan et al. “On hyperparameter tuning in general clustering problems”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2996–3007.
- [96] Lili Blumenberg and Kelly V Ruggles. “Hypercluster: a flexible tool for parallelized unsupervised clustering optimization”. In: *BMC bioinformatics* 21.1 (2020), pp. 1–7.
- [97] Viacheslav Shalamov et al. “Reinforcement-based method for simultaneous clustering algorithm selection and its hyperparameters optimization”. In: *Procedia Computer Science* 136 (2018), pp. 144–153.

Bibliography

- [98] Toon Van Craenendonck and Hendrik Blockeel. “Constraint-based clustering selection”. In: *Machine Learning* 106.9 (2017), pp. 1497–1521.
- [99] Leandro L Minku. “A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation”. In: *Empirical Software Engineering* 24.5 (2019), pp. 3153–3204.
- [100] Ulrike Von Luxburg, Robert C Williamson, and Isabelle Guyon. “Clustering: Science or art?” In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 65–79.
- [101] Lisha Li et al. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.
- [102] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.
- [103] Vipin Kumar and Sonajharia Minz. “Feature selection: a literature review”. In: *SmartCR* 4.3 (2014), pp. 211–229.
- [104] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. “Feature relevance quantification in explainable AI: A causal problem”. In: *International Conference on artificial intelligence and statistics*. PMLR. 2020, pp. 2907–2916.
- [105] Lei Yu and Huan Liu. “Efficient feature selection via analysis of relevance and redundancy”. In: *The Journal of Machine Learning Research* 5 (2004), pp. 1205–1224.
- [106] Bo Tang, Steven Kay, and Haibo He. “Toward optimal feature selection in naive Bayes for text categorization”. In: *IEEE transactions on knowledge and data engineering* 28.9 (2016), pp. 2508–2521.
- [107] Fadi Thabtah et al. “Least Loss: A simplified filter method for feature selection”. In: *Information Sciences* 534 (2020), pp. 1–15.
- [108] Fei Zhao et al. “A filter feature selection algorithm based on mutual information for intrusion detection”. In: *Applied Sciences* 8.9 (2018), p. 1535.
- [109] Wanfu Gao et al. “Feature selection by integrating two groups of feature evaluation criteria”. In: *Expert Systems with Applications* 110 (2018), pp. 11–19.
- [110] Emrah Hancer, Bing Xue, and Mengjie Zhang. “Differential evolution for filter feature selection based on information theory and feature ranking”. In: *Knowledge-Based Systems* 140 (2018), pp. 103–119.
- [111] Marko Robnik-Šikonja and Igor Kononenko. “Theoretical and empirical analysis of ReliefF and RReliefF”. In: *Machine learning* 53.1 (2003), pp. 23–69.
- [112] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
- [113] Magdalene Marinaki and Yannis Marinakis. “An island memetic differential evolution algorithm for the feature selection problem”. In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*. Springer, 2014, pp. 29–42.
- [114] Mahdieh Labani et al. “A novel multivariate filter method for feature selection in text classification problems”. In: *Engineering Applications of Artificial Intelligence* 70 (2018), pp. 25–37.

Bibliography

- [115] Hanchuan Peng, Fuhui Long, and Chris Ding. “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy”. In: *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005), pp. 1226–1238.
- [116] Abdur Rehman et al. “Relative discrimination criterion—A novel feature ranking method for text data”. In: *Expert Systems with Applications* 42.7 (2015), pp. 3670–3681.
- [117] Firuz Kamalov and Fadi Thabtah. “A feature selection method based on ranked vector scores of features for classification”. In: *Annals of Data Science* 4.4 (2017), pp. 483–502.
- [118] J. Ross Quinlan. “Induction of decision trees”. In: *Machine learning* 1.1 (1986), pp. 81–106.
- [119] Huan Liu and Rudy Setiono. “Chi2: Feature selection and discretization of numeric attributes”. In: *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 1995, pp. 388–391.
- [120] Mark Andrew Hall et al. “Correlation-based feature selection for machine learning”. In: (1999).
- [121] Maryam Rahmaninia and Parham Moradi. “OSFSMI: online stream feature selection method based on mutual information”. In: *Applied Soft Computing* 68 (2018), pp. 733–746.
- [122] Pablo A Estévez et al. “Normalized mutual information feature selection”. In: *IEEE Transactions on neural networks* 20.2 (2009), pp. 189–201.
- [123] YongSeog Kim, W Nick Street, and Filippo Menczer. “Evolutionary model selection in unsupervised learning”. In: *Intelligent data analysis* 6.6 (2002), pp. 531–556.
- [124] Mark Hall et al. “The WEKA data mining software: an update”. In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.
- [125] Lior Wolf, Amnon Shashua, and Donald Geman. “Feature Selection for Unsupervised and Supervised Inference: The Emergence of Sparsity in a Weight-Based Approach.” In: *Journal of Machine Learning Research* 6.11 (2005).
- [126] Hong Zeng and Yiu-ming Cheung. “Feature selection and kernel learning for local learning-based clustering”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2010), pp. 1532–1547.
- [127] Zheng Zhao and Huan Liu. “Spectral feature selection for supervised and unsupervised learning”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 1151–1157.
- [128] Isabelle Guyon et al. *Feature extraction: foundations and applications*. Vol. 207. Springer, 2008.
- [129] Zheng Alan Zhao and Huan Liu. *Spectral feature selection for data mining*. Taylor & Francis, 2012.
- [130] Yi Zhang, Chris Ding, and Tao Li. “Gene selection algorithm by combining reliefF and mRMR”. In: *BMC genomics* 9.2 (2008), pp. 1–10.
- [131] Yonghong Peng, Zhiqing Wu, and Jianmin Jiang. “A novel feature selection approach for biomedical data classification”. In: *Journal of Biomedical Informatics* 43.1 (2010), pp. 15–23.

Bibliography

- [132] Ali El Akadi et al. “A two-stage gene selection scheme utilizing MRMR filter and GA wrapper”. In: *Knowledge and Information Systems* 26.3 (2011), pp. 487–500.
- [133] Igor Vainer et al. “Obtaining scalable and accurate classification in large-scale spatio-temporal domains”. In: *Knowledge and information systems* 29.3 (2011), pp. 527–564.
- [134] Eugene Tuv et al. “Feature selection with ensembles, artificial variables, and redundancy elimination”. In: *The Journal of Machine Learning Research* 10 (2009), pp. 1341–1366.
- [135] Yijun Sun. “Iterative RELIEF for feature weighting: algorithms, theories, and applications”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1035–1051.
- [136] Yijun Sun, Sinisa Todorovic, and Steve Goodison. “A feature selection algorithm capable of handling extremely large data dimensionality”. In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM. 2008, pp. 530–540.
- [137] Boris Chidlovskii and Loïc Lecercf. “Scalable feature selection for multi-class problems”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2008, pp. 227–240.
- [138] Steven Loscalzo, Lei Yu, and Chris Ding. “Consensus group stable feature selection”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 567–576.
- [139] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. “Robust feature selection using ensemble feature selection techniques”. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2008, pp. 313–325.
- [140] Diego Peteiro-Barral et al. “Scalability analysis of lter-based methods for feature selection”. In: *Advances in Smart Systems Research* 2.1 (2012), p. 21.
- [141] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC Press, 2018.
- [142] George EP Box and David R Cox. “An analysis of transformations”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 26.2 (1964), pp. 211–243.
- [143] MH Lee, Hossein Javedani Sadaei, and Suhartono. “Improving TAIEX forecasting using fuzzy time series with Box–Cox power transformation”. In: *Journal of Applied Statistics* 40.11 (2013), pp. 2407–2422.
- [144] Jonathan Gillard. “A generalised Box–Cox transformation for the parametric estimation of clinical reference intervals”. In: *Journal of Applied Statistics* 39.10 (2012), pp. 2231–2245.
- [145] Liuquan Sun, Xingwei Tong, and Xian Zhou. “A class of Box-Cox transformation models for recurrent event data”. In: *Lifetime Data Analysis* 17.2 (2011), pp. 280–301.
- [146] Mezbahur Rahman and Larry M Pearson. “Anderson-Darling statistic in estimating the Box-Cox transformation parameter”. In: *Journal of Applied Probability and Statistics* 3.1 (2008), pp. 45–57.
- [147] Jason Osborne. “Improving your data transformations: Applying the Box-Cox transformation”. In: *Practical Assessment, Research, and Evaluation* 15.1 (2010), p. 12.

Bibliography

- [148] Osman Dag, Ozgur Asar, and Ozlem Ilk. "A methodology to implement Box-Cox transformation when no covariate is available". In: *Communications in Statistics-Simulation and Computation* 43.7 (2014), pp. 1740–1759.
- [149] Sarah Stevens et al. "Analysing indicators of performance, satisfaction, or safety using empirical logit transformation". In: *bmj* 352 (2016).
- [150] Gerald A Studebaker. "A" rationalized" arcsine transform". In: *Journal of Speech, Language, and Hearing Research* 28.3 (1985), pp. 455–462.
- [151] Max Kuhn and Kjell Johnson. *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019.
- [152] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.
- [153] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- [154] James V Stone. "Independent component analysis: a tutorial introduction". In: (2004).
- [155] Dominic Langlois, Sylvain Chartier, and Dominique Gosselin. "An introduction to independent component analysis: InfoMax and FastICA algorithms". In: *Tutorials in Quantitative Methods for Psychology* 6.1 (2010), pp. 31–38.
- [156] Nicolas Gillis. "Introduction to nonnegative matrix factorization". In: *arXiv preprint arXiv:1703.00663* (2017).
- [157] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 2012.
- [158] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [159] Sven Serneels, Evert De Nolf, and Pierre J Van Espen. "Spatial sign preprocessing: a simple way to impart moderate robustness to multivariate estimators". In: *Journal of Chemical Information and Modeling* 46.3 (2006), pp. 1402–1409.
- [160] Alan D Fleming et al. "Automated microaneurysm detection using local contrast normalization and local vessel detection". In: *IEEE transactions on medical imaging* 25.9 (2006), pp. 1223–1232.
- [161] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.
- [162] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.
- [163] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "' Why should i trust you?' Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.
- [164] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.
- [165] Wojciech Samek et al. *Explainable AI: interpreting, explaining and visualizing deep learning*. Vol. 11700. Springer Nature, 2019.

Bibliography

- [166] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *International conference on machine learning*. PMLR. 2016, pp. 478–487.
- [167] Bo Yang et al. “Towards k-means-friendly spaces: Simultaneous deep learning and clustering”. In: *international conference on machine learning*. PMLR. 2017, pp. 3861–3870.
- [168] Mathilde Caron et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 132–149.
- [169] Kamran Ghasedi Dizaji et al. “Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5736–5745.
- [170] Xifeng Guo et al. “Deep clustering with convolutional autoencoders”. In: *International conference on neural information processing*. Springer. 2017, pp. 373–382.
- [171] Xi Peng et al. “k-meansnet: When k-means meets differentiable programming”. In: *arXiv preprint arXiv:1808.07292* (2018).
- [172] Martin HC Law, Mario AT Figueiredo, and Anil K Jain. “Simultaneous feature selection and clustering using mixture models”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.9 (2004), pp. 1154–1166.
- [173] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. “On clustering validation techniques”. In: *Journal of intelligent information systems* 17.2 (2001), pp. 107–145.
- [174] Marina Meila. “How to tell when a clustering is (approximately) correct using convex relaxations”. In: *Advances in Neural Information Processing Systems* 31 (2018).
- [175] Tilman Lange et al. “Stability-based validation of clustering solutions”. In: *Neural computation* 16.6 (2004), pp. 1299–1323.
- [176] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge, 2008.
- [177] Tauno Metsalu and Jaak Vilo. “ClustVis: a web tool for visualizing clustering of multivariate data using Principal Component Analysis and heatmap”. In: *Nucleic acids research* 43.W1 (2015), W566–W570.
- [178] Michael Kern et al. “Interactive visual exploration and refinement of cluster assignments”. In: *BMC bioinformatics* 18.1 (2017), pp. 1–13.
- [179] Katja Hansen et al. “Visual Interpretation of Kernel-based prediction models”. In: *Molecular Informatics* 30.9 (2011), pp. 817–826.
- [180] Huan Liu and Lei Yu. “Toward integrating feature selection algorithms for classification and clustering”. In: *IEEE Transactions on knowledge and data engineering* 17.4 (2005), pp. 491–502.
- [181] Joseph Lee Rodgers and W Alan Nicewander. “Thirteen ways to look at the correlation coefficient”. In: *The American Statistician* 42.1 (1988), pp. 59–66.
- [182] MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.

Bibliography

- [183] Peter E Hart, David G Stork, and Richard O Duda. *Pattern classification*. Wiley Hoboken, 2000.
- [184] Mingjie Qian and Chengxiang Zhai. “Robust unsupervised feature selection”. In: *Twenty-third international joint conference on artificial intelligence*. Citeseer. 2013.
- [185] HS Shukla, Narendra Kumar, and RP Tripathi. “Gaussian noise filtering techniques using new median filter”. In: *International Journal of Computer Applications* 95.12 (2014).
- [186] Xiaofei He, Deng Cai, and Partha Niyogi. “Laplacian score for feature selection”. In: *Advances in neural information processing systems* 18 (2005).
- [187] Isabelle Guyon and André Elisseeff. “An introduction to variable and feature selection”. In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.
- [188] Jennifer G Dy and Carla E Brodley. “Feature selection for unsupervised learning”. In: *Journal of machine learning research* 5.Aug (2004), pp. 845–889.
- [189] Yijuan Lu et al. “Feature selection using principal feature analysis”. In: *Proceedings of the 15th ACM international conference on Multimedia*. 2007, pp. 301–304.
- [190] George P McCabe. “Principal variables”. In: *Technometrics* 26.2 (1984), pp. 137–144.
- [191] Deng Cai, Chiyuan Zhang, and Xiaofei He. “Unsupervised feature selection for multi-cluster data”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 333–342.
- [192] Aapo Hyvärinen and Erkki Oja. “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5 (2000), pp. 411–430.
- [193] Jonathon Shlens. “A tutorial on principal component analysis”. In: *arXiv preprint arXiv:1404.1100* (2014).
- [194] Vicente Zarzoso, Pierre Comon, and Mariem Kallel. “How fast is FastICA?” In: *2006 14th European Signal Processing Conference*. IEEE. 2006, pp. 1–5.
- [195] Alan E Hendrickson and Paul Owen White. “Promax: A quick method for rotation to oblique simple structure”. In: *British journal of statistical psychology* 17.1 (1964), pp. 65–70.
- [196] Henry F Kaiser. “The varimax criterion for analytic rotation in factor analysis”. In: *Psychometrika* 23.3 (1958), pp. 187–200.
- [197] Ron Mancini. *Op amps for everyone: design reference*. Newnes, 2003.
- [198] Kohei Arai and Ali Ridho Barakbah. “Hierarchical K-means: an algorithm for centroids initialization for K-means”. In: *Reports of the Faculty of Science and Engineering* 36.1 (2007), pp. 25–31.
- [199] Xihong Cui et al. “GPR-Based Automatic Identification of Root Zones of Influence Using HDBSCAN”. In: *Remote Sensing* 13.6 (2021), p. 1227.
- [200] Marco Rovere et al. “Clue: A fast parallel clustering algorithm for high granularity calorimeters in high-energy physics”. In: *Frontiers in big Data* 3 (2020), p. 41.
- [201] Emmanuelle A Marquis et al. “On the use of density-based algorithms for the analysis of solute clustering in atom probe tomography data”. In: *Proceedings of the 18th International Conference on Environmental Degradation of Materials in Nuclear Power Systems–Water Reactors*. Springer. 2019, pp. 2097–2113.

Bibliography

- [202] Mehjabin Khatoon and W Aisha Banu. “An efficient method to detect communities in social networks using DBSCAN algorithm”. In: *Social Network Analysis and Mining* 9.1 (2019), pp. 1–12.
- [203] Carly GK Ziegler et al. “SARS-CoV-2 receptor ACE2 is an interferon-stimulated gene in human airway epithelial cells and is detected in specific cell subsets across tissues”. In: *Cell* 181.5 (2020), pp. 1016–1035.
- [204] Conor Fahy and Shengxiang Yang. “Finding and tracking multi-density clusters in online dynamic data streams”. In: *IEEE Transactions on Big Data* (2019).
- [205] Sean M Carroll. *Spacetime and geometry*. Cambridge University Press, 2019.
- [206] Cornelius Von Westenholz. *Differential forms in mathematical physics*. Elsevier, 2009.
- [207] Charalampos Konstantopoulos. “A parallel algorithm for motion estimation in video coding using the bilinear transformation”. In: *SpringerPlus* 4.1 (2015), pp. 1–20.
- [208] Na Xiao et al. “A Novel Clustering Algorithm based on Directional Propagation of Cluster Labels”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.
- [209] Claudia Malzer and Marcus Baum. “Hdbscan (): An alternative cluster extraction method for HDBSCAN”. In: *CoRR, abs/1911.02282* (2019).
- [210] Peng Liu, Dong Zhou, and Naijun Wu. “VDBSCAN: varied density based spatial clustering of applications with noise”. In: *2007 International conference on service systems and service management*. IEEE. 2007, pp. 1–4.
- [211] Shimei Wang, Yun Liu, and Bo Shen. “MDBSCAN: multi-level density based spatial clustering of applications with noise”. In: *Proceedings of the The 11th International Knowledge Management in Organizations Conference on The changing face of Knowledge Management Impacting Society*. 2016, pp. 1–5.
- [212] Alex Rodriguez and Alessandro Laio. “Clustering by fast search and find of density peaks”. In: *science* 344.6191 (2014), pp. 1492–1496.
- [213] Bo Chen et al. “Local contrast as an effective means to robust clustering against varying densities”. In: *Machine Learning* 107.8 (2018), pp. 1621–1645.
- [214] Abdulrahman Lotfi, Parham Moradi, and Hamid Beigy. “Density peaks clustering based on density backbone and fuzzy neighborhood”. In: *Pattern Recognition* 107 (2020), p. 107449.
- [215] Ye Zhu, Kai Ming Ting, and Mark J Carman. “Density-ratio based clustering for discovering clusters with varying densities”. In: *Pattern Recognition* 60 (2016), pp. 983–997.
- [216] Ye Zhu, Kai Ming Ting, and Maia Angelova. “A distance scaling method to improve density-based clustering”. In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2018, pp. 389–400.
- [217] Ye Zhu et al. “CDF Transform-and-Shift: An effective way to deal with datasets of inhomogeneous cluster densities”. In: *Pattern Recognition* 117 (2021), p. 107977.
- [218] Laura MP Mariño and Francisco de AT de Carvalho. “A new batch SOM algorithm for relational data with weighted medoids”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

Bibliography

- [219] William Wells Adams et al. *An introduction to Grobner bases*. 3. American Mathematical Soc., 1994.
- [220] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.
- [221] Shu-Chuan Chu. *Improved Clustering and Soft Computing Algorithms*. Flinders University of South Australia, School of Informatics and Engineering, 2004.
- [222] Marcio Gameiro et al. “A topological measurement of protein compressibility”. In: *Japan Journal of Industrial and Applied Mathematics* 32.1 (2015), pp. 1–17.
- [223] Gunnar Carlsson et al. “On the local behavior of spaces of natural images”. In: *International journal of computer vision* 76.1 (2008), pp. 1–12.
- [224] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. “Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival”. In: *Proceedings of the National Academy of Sciences* 108.17 (2011), pp. 7265–7270.
- [225] Hans-Peter Kriegel et al. “Density-based clustering”. In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1.3 (2011), pp. 231–240.
- [226] Yinghua Lv et al. “An efficient and scalable density-based clustering algorithm for datasets with complex structures”. In: *Neurocomputing* 171 (2016), pp. 9–22.
- [227] Avory Bryant and Krzysztof Cios. “RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.6 (2017), pp. 1109–1121.
- [228] Yifeng Lu et al. “k-Nearest Neighbor based Clustering with Shape Alternation Adaptivity”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.
- [229] Salem Alelyani, Jiliang Tang, and Huan Liu. “Feature selection for clustering: A review”. In: *Data Clustering* (2018), pp. 29–60.
- [230] Carl Rasmussen. “The infinite Gaussian mixture model”. In: *Advances in neural information processing systems* 12 (1999).
- [231] Ian T Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.
- [232] Marley W Watkins. “Exploratory factor analysis: A guide to best practice”. In: *Journal of Black Psychology* 44.3 (2018), pp. 219–246.
- [233] Alexander Hinneburg, Daniel A Keim, and Markus Wawryniuk. “HD-Eye: Visual mining of high-dimensional data”. In: *IEEE Computer Graphics and Applications* 19.5 (1999), pp. 22–31.
- [234] Charu C Aggarwal. “A human-computer interactive method for projected clustering”. In: *IEEE transactions on knowledge and data engineering* 16.4 (2004), pp. 448–460.
- [235] David Cohn, Rich Caruana, and Andrew McCallum. “Semi-supervised clustering with user feedback”. In: *Constrained Clustering: Advances in Algorithms, Theory, and Applications* 4.1 (2003), pp. 17–32.
- [236] Zhengdong Lu and Todd K Leen. “Pairwise constraints as priors in probabilistic clustering”. In: *Basu et al.(2008)* (2008), pp. 59–90.

Bibliography

- [237] Pek Y Lum et al. “Extracting insights from the shape of complex data using topology”. In: *Scientific reports* 3.1 (2013), pp. 1–8.
- [238] Gurjeet Singh, Facundo Mémoli, Gunnar E Carlsson, et al. “Topological methods for the analysis of high dimensional data sets and 3d object recognition.” In: *PBG@Eurographics* 2 (2007).
- [239] Hujun Yin. “The self-organizing maps: background, theories, extensions and applications”. In: *Computational intelligence: A compendium*. Springer, 2008, pp. 715–762.
- [240] E Arsuaga Uriarte and F Díaz Martín. “Topology preservation in SOM”. In: *International journal of applied mathematics and computer sciences* 1.1 (2005), pp. 19–22.
- [241] Jing Tian, Michael H Azarian, and Michael Pecht. “Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm”. In: *PHM Society European Conference*. Vol. 2. 1. 2014.
- [242] Marian Pena, Wesam Barbakh, and Colin Fyfe. “Topology-preserving mappings for data visualisation”. In: *Principal Manifolds for Data Visualization and Dimension Reduction*. Springer, 2008, pp. 131–150.
- [243] Richard Hugh Moulton and Jakub Zgraja. “The Wilderness Area Data Set: Adapting the Covertypes data set for unsupervised learning”. In: *arXiv preprint arXiv:1901.11040* (2019).
- [244] David H Wolpert, William G Macready, et al. *No free lunch theorems for search*. Tech. rep. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.
- [245] Gil Press. *Cleaning Big Data: Most time-consuming, least enjoyable data science task, survey says*. 2021. URL: <https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=45c424a96f63>.
- [246] Fatma Ozge Ozkok and Mete Celik. “A new approach to determine Eps parameter of DBSCAN algorithm”. In: *International Journal of Intelligent Systems and Applications in Engineering* 5.4 (2017), pp. 247–251.
- [247] Wei-Tung Wang et al. “Adaptive density-based spatial clustering of applications with noise (DBSCAN) according to data”. In: *2015 International Conference on Machine Learning and Cybernetics (ICMLC)*. Vol. 1. IEEE. 2015, pp. 445–451.
- [248] Alexander Dockhorn, Christian Braune, and Rudolf Kruse. “An alternating optimization approach based on hierarchical adaptations of dbscan”. In: *2015 IEEE Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 749–755.
- [249] Xiaoming Chen et al. “APSCAN: A parameter free algorithm for clustering”. In: *Pattern Recognition Letters* 32.7 (2011), pp. 973–986.
- [250] Jian Hou, Huijun Gao, and Xuelong Li. “DSets-DBSCAN: A parameter-free clustering algorithm”. In: *IEEE Transactions on Image Processing* 25.7 (2016), pp. 3182–3193.
- [251] Amin Karami and Ronnie Johansson. “Choosing DBSCAN parameters automatically using differential evolution”. In: *International Journal of Computer Applications* 91.7 (2014), pp. 1–11.

Bibliography

- [252] Huang Darong and Wang Peng. “Grid-based DBSCAN algorithm with referential parameters”. In: *Physics Procedia* 24 (2012), pp. 1166–1170.
- [253] Abir Smiti and Zied Elouedi. “Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques”. In: *2012 IEEE 16th international conference on intelligent engineering systems (INES)*. IEEE. 2012, pp. 573–578.
- [254] G Phanendra Babu and M Narasimha Murty. “Simulated annealing for selecting optimal initial seeds in the k-means algorithm”. In: *Indian Journal of Pure and Applied Mathematics* 25.1-2 (1994), pp. 85–94.
- [255] Amin Karami and Manel Guerrero-Zapata. “A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks”. In: *Neurocomputing* 149 (2015), pp. 1253–1269.
- [256] George E Tsekouras. “A simple and effective algorithm for implementing particle swarm optimization in RBF network’s design using input-output fuzzy clustering”. In: *Neurocomputing* 108 (2013), pp. 36–44.
- [257] Junyan Chen. “Hybrid clustering algorithm based on PSO with the multidimensional asynchronism and stochastic disturbance method”. In: *Journal of Theoretical and Applied Information Technology* 46.1 (2012), pp. 434–440.
- [258] Lin Li and Suhua Liu. “Wheat cultivar classifications based on tabu search and fuzzy c-means clustering algorithm”. In: *2012 Fourth International Conference on Computational and Information Sciences*. IEEE. 2012, pp. 493–496.
- [259] Hong-Bing Xu, Hou-Jun Wang, and Chun-Guang Li. “Fuzzy tabu search method for the clustering problem”. In: *Proceedings. International Conference on Machine Learning and Cybernetics*. Vol. 2. IEEE. 2002, pp. 876–880.
- [260] Yangyang Li et al. “A spectral clustering-based adaptive hybrid multi-objective harmony search algorithm for community detection”. In: *2012 IEEE Congress on Evolutionary Computation*. IEEE. 2012, pp. 1–8.
- [261] Carlos Cobos et al. “Web document clustering based on global-best harmony search, K-means, frequent term sets and Bayesian information criterion”. In: *IEEE congress on evolutionary computation*. IEEE. 2010, pp. 1–8.
- [262] Liang Li et al. “A combinatorial search method based on harmony search algorithm and particle swarm optimization in slope stability analysis”. In: *2009 International Conference on Computational Intelligence and Software Engineering*. IEEE. 2009, pp. 1–4.
- [263] Emrah Hancer, Celal Ozturk, and Dervis Karaboga. “Artificial bee colony based image clustering method”. In: *2012 IEEE congress on evolutionary computation*. IEEE. 2012, pp. 1–5.
- [264] Dervis Karaboga, Selcuk Okdem, and Celal Ozturk. “Cluster based wireless sensor network routing using artificial bee colony algorithm”. In: *Wireless Networks* 18.7 (2012), pp. 847–860.
- [265] Yannis Marinakis, Magdalene Marinaki, and Nikolaos Matsatsinis. “A hybrid discrete artificial bee colony-GRASP algorithm for clustering”. In: *2009 International Conference on Computers & Industrial Engineering*. IEEE. 2009, pp. 548–553.

Bibliography

- [266] Bo Zhao et al. “Image segmentation based on ant colony optimization and K-means clustering”. In: *2007 IEEE International Conference on Automation and Logistics*. IEEE. 2007, pp. 459–463.
- [267] Yanfang Han and Pengfei Shi. “An improved ant colony algorithm for fuzzy clustering in image segmentation”. In: *Neurocomputing* 70.4-6 (2007), pp. 665–671.
- [268] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
- [269] Stanisaw Sieniutycz and Jacek Jeowski. “1-brief review of static optimization methods”. In: *Energy optimization in process systems and fuel cells* (2018), pp. 1–43.
- [270] Yang Sun, Lingbo Zhang, and Xingsheng Gu. “A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems”. In: *Neurocomputing* 98 (2012), pp. 76–89.
- [271] Hongfang Zhou, Peng Wang, and Hongyan Li. “Research on adaptive parameters determination in DBSCAN algorithm”. In: *Journal of Information & Computational Science* 9.7 (2012).
- [272] Jörg Sander et al. “Density-based clustering in spatial databases: The algorithm gdbscan and its applications”. In: *Data mining and knowledge discovery* 2.2 (1998), pp. 169–194.
- [273] Shiming Xiang, Feiping Nie, and Changshui Zhang. “Learning a Mahalanobis distance metric for data clustering and classification”. In: *Pattern recognition* 41.12 (2008), pp. 3600–3612.
- [274] Fei Wang et al. “An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity”. In: *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2017, pp. 291–305.