Identification of Data Structure with Machine Learning: From Fisher to Bayesian networks

Raúl V. Casaña-Eslava

A thesis submitted in partial fulfillment for the requirements of
Liverpool John Moores University
for the degree of Doctor of Philosophy

December 2018

Declaration of Authorship

I, Raúl V. Casaña-Eslava, declare that this thesis titled, 'Identification of Data Structure with Machine Learning: From Fisher to Bayesian networks' and the work presented in it are my own. I confirm that:

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

Signed:

Date:

"When it is not in our power to determine what is true, we ought to act in accordance with what is most probable."

René Descartes.

Liverpool John Moores University

Abstract

Faculty of Engineering and Technology
Department of Applied Mathematics

Doctor of Philosophy

by Raúl V. Casaña-Eslava

This thesis proposes a theoretical framework to thoroughly analyse the structure of a dataset in terms of a) metric, b) density and c) feature associations. To look into the first aspect, Fisher's metric learning algorithms are the foundations of a novel manifold based on the information and complexity of a classification model. When looking at the density aspect, the Probabilistic Quantum clustering, a Bayesian version of the original Quantum Clustering is proposed. The clustering results will depend on local density variations, which is a desired feature when dealing with heteroscedastic data. To address the third aspect, the constraint-based PC-algorithm is the starting point of many structure learning algorithms, it is focused on finding feature associations by means of conditional independent tests. This is then used to select Bayesian networks, based on a regularized likelihood score.

These three topics of data structure analysis were fully tested with synthetic data examples and real cases, which allowed us to unravel and discuss the advantages and limitations of these algorithms. One of the biggest challenges encountered was related to the application of these methods to a Big Data dataset that was analysed within the framework of a collaboration with a large UK retailer, where the interest was in the identification of the data structure underlying customer shopping baskets.

Acknowledgements

In reference to who encouraged me to start this Ph.D., I would like to start by thanking Professor José D. Martín-Guerrero, he was the one who introduced me to the fascinating world of machine learning (in this part I also include Dr. Antonio Serrano-Lopez). José was also the one who trust me and gave me the opportunity to research in the IDAL group at UV, and finally he was the one who recommended me to go to Liverpool to continue my studies with the research team of Professor Paulo Lisboa at LJMU.

Now, in reference to the doctorate itself, of course I want to greatly thank my supervisors Dr. Ian H. Jarman, Professor Paulo Lisboa and Dr. Sandra Ortega-Martorell, for the help, motivation and guidance they have offered me. During these years I have considered myself very fortunate to have had them as supervisors, teachers and friends. And definitely without them, this work would not have been possible. I could talk at length about our interesting weekly meetings, where each one taught me to think and to approach new research problems from different points of view, forming, in short, a very balanced team that has made me grow as a researcher and as a person.

Finally, I wanted to thank the support and friendship of my department colleagues Aday, Victor, Wajdi and Akeel. And last but not least, the love and unconditional support I received in the distance of my family and Irene, who struggled with a long-distance relationship.

Contents

List of Figures

List of Tables

Acronyms

ANLL	**A**verage **N**egative **L**og-**L**ikelihood
API	**A**pplication **P**rogramming **I**nterface
APSP	**A**ll **P**airs **S**hortest **P**ath
AWS	**A**mazon **W**eb **S**ervices
BN	**B**ayesian **N**etwork
BIC	**B**ayesian **I**nformation **C**riterion
CI-map	**C**onditional **I**ndependence map
CPU	**C**entral **P**rocessing **U**nit
cMDS	**c**lassical **M**ulti**D**imensional **S**caling
DAG	**D**irected **A**cyclic **G**raph
DANN	**D**iscriminant **A**daptive **N**earest **N**eighbour
DBN	**D**ynamic **B**ayesian **N**etwork
DBN	**D**ynamic **B**ayesian **N**etwork
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
FDR	**F**alse **D**iscovery **R**ate
FNR	**F**alse **N**egative **R**eduction
FI	**F**isher **I**nformation
FIN	**F**isher **I**nformation **N**etwork
GPGPU	**G**eneral-**P**urpose **C**omputing on **G**raphics **P**rocessing **U**nits
GSDM	**G**lobal **S**imilarity **D**istance **M**etric
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
HPC	**H**igh **P**erformance **C**omputer
JS	**J**accard **S**core

JVM	**J**ava **V**irtual **M**achine
KL-div	**K**ullback-**L**eibler divergence
KNN	**K**-**N**earest **N**eighbours
L-BFGS	**L**imited-memory **B**royden-**F**letcher-**G**oldfarb-**S**hanno
LDA	**L**inear **D**iscriminant **A**nalysis
LLE	**L**ocal **L**inear **E**mbedding
LMNN	**L**arge **M**argin **N**earest **N**eighbour
MDS	**M**ulti**D**imensional **S**caling
MGM	**M**ixture of **G**aussian **M**odel
MLP	**M**ulti-**L**ayer **P**erceptron
ML	**M**achine **L**earning
NCA	**N**eighbourhood **C**omponent **A**nalysis
PC-algorithm	**P**eter-**C**lark algorithm
PCA	**P**rincipal **C**omponent **A**nalysis
PCoA	**P**rincipal **C**oordinate **A**nalysis
PDAG	**P**artially **D**irected **A**cyclic **G**raph
PHE	**P**ublic **H**ealth **E**ngland
PIC	**P**ower **I**teration **C**lustering
PQC	**P**robabilistic **Q**uantum **C**lustering
QC	**Q**uantum **C**lustering
RCA	**R**elevant **C**omponent **A**nalysis
RDD	**R**esilent **D**istributed **D**ataset
RFM	**R**ecency **F**requency **M**onetary
RAM	**R**andom **A**ccess **M**emory
SeCo	**S**eparation-**C**oncordance
SGD	**S**tochastic **G**radient **D**escent
SL-approach	**S**traigh **L**ine approach
SQL	**S**tructured **Q**uery **L**anguage
SSSP	**S**ingle **S**ource **S**hortest **P**ath
t-SNE	**t**-distributed **S**tochastic **N**eighbour **E**mbedding
TSF	**T**he **S**trongest **F**irst
TWF	**T**he **W**eakest **F**irst
UDF	**U**ser **D**efined **F**unction

UK	**U**nited **K**ingdom
YARN	**Y**et **A**nother **R**esource **N**egociator
ZMI	**Z**ernike **M**oment **I**nvariant

To my father...

Chapter 1

Introduction

This thesis can be summarised as a search for data structure using three different perspectives that belong to distinct machine learning fields: metric learning, unsupervised clustering and structure learning. The structure analysis is driven through a pipeline of methods divided in three parts:

- The first part focuses on the computation of a new metric that redefines the distances between observations creating a new data space with different neighbourhoods.
- The second part applies clustering techniques to find new clusters formed by new metric.
- The third part looks for feature associations within each cluster defined in the previous part, assuming that each cluster may be driven by different feature relationships.

This chapter introduces a global perspective on this work, giving additional details about the high level pipeline overview in the next section 1.1. Then, it is followed by the sections: context 1.2, rationale 1.3, algorithm pipeline schema 1.4, aims and objectives 1.5, novel contributions 1.6, limitations 1.7, list of publications 1.8 and thesis layout 1.9.

1.1 High level overview

The main goal of the pipeline is to discover new groups of similar data (clusters) when the metric that defines the similarity of the observations is changed, and where these

new clusters would be more difficult to detect with the original Euclidean metric. The proposed metric is derived from the Fisher metric, which it is not Euclidean and depends on additional external data labels, henceforth identified as *class labels*. The class labels give additional properties to each observation and the metric will tend to separate as much as possible the observations with different class labels, and to group together observations with the same class label. However, although the distance between observations is affected by the class label to which the observations belong, the main contribution to the distance measurement still depends on the feature similarities.

Therefore, the idea of changing the metric is for smoothly restructuring the data space in such a way that a new data neighbourhood is formed and new clusters can be identified, where the new space offers a continuous granularity from one class region to another. If a new class label set is given, recomputing the metric again will produce a different data distribution.

Alternatively, a completely different approach could be taken, for instance, directly separating the observations by class labels. But doing this, the distance as a concept of feature similarities between observations are not taken into account, i.e. the observations are separated arbitrarily by its class labels, where the class labels do not have to be related with data features. It is possible that without the change of metric, the new detected clusters could be overlapped in the original space.

This is the first part of the analysis, which is focused on the identification of similarities between observations when the metric is changed. The second part of the analysis is focused on the identification of associations between features.

Roughly speaking two associated features can be understood when they are correlated, if one feature varies the other will vary in a corresponding manner. The absence of correlation indicates these features are independent. The measure of association between two features is computed through independence tests, and can be extended to all features creating a map of conditional independence tests, henceforth *Conditional Independence-Maps* (CI-Maps). These maps can be applied on each cluster, highlighting the feature associations intra-cluster as a characteristic cluster driver (or behaviour understood as a pattern of CI-Maps), where the CI-Maps are indirectly dependent on the class labels (because the clusters depend on them). Without considering the cluster stratification in the CI-Maps, the multiple drivers tend to be masked when the feature associations are analysed on the whole data, where in general the global behaviour is a superposition of multiple drivers. For that reason, the idea of building CI-Maps by cluster is to discover different drivers that are aggregated in the same data space.

In terms of pipeline inputs and outputs, the expected input of the pipeline is a dataset with class labels that will be used to compute the new metric. For the outputs, there are three parts in the pipeline:

1. The first output is related to metric learning, this part can be catalogued as semi-supervised because the metric depends on the class labels. The new metric based on the Fisher metric defines a Riemannian manifold, called henceforth *Fisher manifold*. Under this space, new pairwise distances between observations can be estimated, where they differ from the original Euclidean distances. These distances can be represented in a triangular adjacency matrix of $0.5N(N-1)$ elements, where the N observations are represented in N rows and N columns. This matrix contains the necessary information to embed the Fisher manifold in a new low-dimensional Euclidean space. **The embedded Fisher manifold is the first pipeline output**, where the observations are represented by coordinates instead of pairwise distances.

2. The second output is related to unsupervised clustering. **The output is a set of labels identifying clusters**. Two different branches of clustering techniques have been chosen: the first branch is based on spectral clustering that works directly on the distance adjacency matrix, while the second branch is based on projective methods that works on the embedded Fisher manifold. Depending on the structure of data within and between the new clusters, this informs whether the focus is should be on segmentation or density discrimination. Plotting the embedded Fisher manifold will help to decide which method is more appropriate.

3. The third output is related to structure learning, which is also an unsupervised methodology. **The output is a set of CI-Maps of the relevant clusters identified in the previous part**. The CI-Maps can be represented as a network graph with nodes and edges, where the nodes represent the features and the edges represent associations between features (nodes). An additional output are the Bayesian networks derived from the CI-Maps. In general, this part of the analysis is optional and relies on the nature of the input dataset features, and whether it makes sense to look for feature associations.

A summary of the thesis pipeline in a few words could be: *the pipeline starts changing the metric to define a new neighbourhood of the observations, then there is a process for identifying clusters in this new neighbourhood, and finally there is a process to explore feature associations in each cluster.*

1.2 Context

This thesis presents a novel theoretical framework that can be useful in carrying out detailed and thorough analyses of data structure. Specifically, three features of particular relevance are investigated: metric, density and structure.

Metric learning algorithms based on Fisher information propose a semi-supervised approach to produce manifolds based on class labels and the internal characteristics of classification models. This part of the research was carried out in the context of a collaboration with a large UK retailer, interested in a scalable version of Fisher Information Networks (FIN) [2–4]. Thus, a customer segmentation based on shopping baskets of customer transactional data could be performed. The company was also interested in a Big Data framework based on Hadoop [5] and Spark [6]. Once the code was developed in Spark, the algorithms were tested with real data and a procedure was designed to provide insights into the data structure.

Density is analysed by algorithms based on Quantum Clustering (QC) - a quantum inspired unsupervised algorithm for clustering non-spherical distributions. This is an important component of the overall objectives of the thesis, given the capability of this algorithm of finding particular profiles and behaviours that are not usually detected by classical algorithms. One of the novelties of this thesis is the proposed Bayesian version of the QC method, which, in addition to the intrinsic probabilistic interpretation, it is used to detect outliers and assess the free model hyper-parameters in a completely unsupervised way, using a likelihood score function based on cluster membership. The latter is of particular importance as it has the potential to become the basis of an objective evaluation of unsupervised algorithms.

Structure learning is the goal of constraint-based algorithms that can build Conditional Independence Maps (CI-map) [7] with the purpose of building feasible Bayesian Networks (BNs) from data. Part of the initial work that should be highlighted was based on the stabilization of the PC-algorithm [8], analysing different algorithm policies and the influence of the node ordering in the BN construction.

The stages of processing starts with metric learning algorithms that create a Riemannian manifold with the simplest manifold structure for encoding classifier complexity. Then, either spectral clustering is used for community detection, thus converting the Riemannian space into a similarity network based on a Gaussian-kernel distance, or else a Euclidean embedding is used for applying projective clustering methods, like the density-based Quantum Clustering. In both cases the purpose is to find relevant clusters in the Fisher manifold. Then finally, structure learning algorithms are applied over the

clusters to find association maps among the features, where each cluster is likely to have a different structure.

1.3 Rationale

1.3.1 Metric Learning

Following the pipeline order, the main idea of metric learning is how the metric of the data Euclidean space can be changed, in such a way that the new metric reflects a function of given class labels. Changing the metric modifies the pairwise distances between observations, and therefore the data structure. In this new data structure, the aim is to find relevant clusters that provide insights about the given class labels. As mentioned before, if new labels are provided, recomputing the metric, creates a new different data structure.

A suitable metric for this purpose is Fisher Information (FI), which is a Riemannian metric defined on a smooth statistical manifold and is related to the Kullback-Leibler (KL) divergence [9] at neighbouring points in the parameter space using a second order Taylor expansion. Although the FI metric is based on the parameter space of the statistical manifold, it can also be derived analogously using the input data space, as shown in [10], given a posterior probability $P(c|\mathbf{X})$ fitted to the input data and being c the class labels.

A good candidate for the posterior probability model, which has to be up to second order differentiable because of the Hessian matrix of the KL divergence, is a Multilayer Perceptron (MLP), a discriminative model to classify class labels given the input data.

Computing pairwise distances with the FI metric is not straightforward because the metric varies with the input space, therefore it is necessary to sample the metric across the path and use approximations for path integrals and geodesic distances following the shortest path between nodes. This operation could be very expensive if the sample size (N) increases more than several thousands, having $N(N-1)/2$ different pairwise distances. This is mainly due to the sample size, but the number of features and the number of class labels can also have a strong impact on the runtime.

Previous works related to the methodology [2–4] were implemented in Matlab [11] using a single machine, without parallel computing. The runtime can be excessive, for instance, it can be longer than 10 days for a sample of 7500 observations in a standard computer (Intel Core I7 processor, 8Gb RAM memory).

To be able to analyse the Big Data dataset from the UK retailer company using Fisher Information networks, a scalable version of the algorithm, developed in a framework suitable for production, and able to deal with bigger datasets was required. Of the different options to design a scalable algorithm described later in Section 3.3, the Apache Spark framework under the Hadoop ecosystem was chosen, using a scalable computer cluster as the distributed computing paradigm.

Therefore, scalability was the main requirement of the metric learning block to speed up the pairwise distances runtime, which at least depends on $O(N^2)$.

1.3.2 Unsupervised Clustering

Once the pairwise distances based on the FI metric are obtained, the next task in the procedure is to analyse the data structure in the new Riemannian space.

One option described in [10] is to find communities using spectral clustering algorithms, like Newman's modularity optimization algorithm [12]. This step requires the transformation of pairwise distances into a similarity network, where a Gaussian kernel can be used to obtain similarity measures. The advantage is that Newman's algorithm can deal with big data sample sizes. The disadvantage is that the Gaussian kernel introduces a local network hyper-parameter that acts like a length scale which has to be tuned in. This length scale controls the amount of communities (clusters), and it can be estimated by measuring the Fisher network capability of reproducing the class-membership probabilities given by the original estimator (MLP model), for instance, measuring the KL divergence of both models. The length scale can be estimated using heuristic rules based on percentages of average Fisher pairwise distances within the same class-membership. One of the drawbacks of spectral clustering is that it is hard to identify and visualize the data structure without additional methods.

Another option to find clusters is to apply a Euclidean embedding of the Riemannian manifold, and then apply usual clustering methods in a projective space. There are several methods in manifold learning that can be useful to visualize and represent the data structure of a Riemannian manifold in a low-dimensional embedding, for instance: Sammon mapping [13], Isomap [1], Minimap [14], local linear embedding (LLE) [15], classical Multidimensional Scaling (cMDS) [16], or the non-linear t-SNE [17].

From all of these methods, cMDS was chosen, which is a linear and global structure preserving method. One of the main reasons for choosing cMDS is because the eigenvector decomposition allows an estimate of the relative contribution of each eigenvector by the cumulative sum of its eigenvalues, in such a way that only the main eigenvectors are

kept, giving at the same time information about how many dimensions are required to embed the Riemmanian data structure.

Once the Fisher Information space is embedded in a Euclidean space, projective clustering methods can be applied to find relevant clusters. Here, one may observe two opposing situations: The first when the embedded data presents a uniform distribution without local density changes. In reality, there are no clear clusters and the community finding task is a segmentation task. In this case, K-means can segment the data just as good as the spectral clustering methods, and the number of clusters would depend on the desired granularity for the problem at hand. To reinforce the consistency of the number of clusters, K-means was used with the SeCo framework [18], a method to determine the number of clusters by measuring the separation and concordance of different K-Means initializations. Cases of segmentation without clear clusters usually happens when the data is very noisy, or when the MLP cannot properly discriminate the class labels, in which case the low MLP performance is reflected in the FI metric.

The second case is when the embedded Fisher manifold is highly structured or presents high local density variations, mainly due to a very good performance in the MLP. Here, a density-based clustering method able to find non-spherical cluster distributions or discriminate clusters by local density changes is needed. Initially the method of Quantum Clustering (QC) [19] was considered, which outperforms K-means in clustering non-spherical distributions [20]. However, QC had the problem of selecting the appropriate length scale in an unsupervised way. The SeCo framework to the QC in [21] was adapted, measuring the concordance and the range of the QC length scale when sampling QC solutions. This work provided an empirical procedure to estimate the length scale, but the original QC model still had theoretical limitations related to heteroscedasticity and local density variations.

An important part of the thesis consisted then in redesigning QC, providing it with a probability framework able to select the appropriate hyper-parameters in an unsupervised way, through a score function based on a likelihood of cluster membership. The proposed method, called Probabilistic Quantum Clustering (PQC), uses a variable length scale fitted to the local manifold that addresses the heteroscedastic problems. Therefore PQC solves the unsupervised clustering problem and highlights the underlying structure of the Fisher manifold assessing PQC solutions (different hyper-parameters) with the likelihood function.

1.3.3 Structure learning

This section investigates how to build feasible maps of feature relationships from the data. The starting point is the constraint-based PC-algorithm [8], which is used to create a multivariate correlation model represented as a Conditional Independence Map (CI-map), or graphical model that reflects the statistical associations between features. This takes into account multiple levels of conditioning and therefore removes spurious associations i.e. controls for False Positives.

Assuming the hypothesis that the CI-map of the whole data is composed by a mixture of underlying drivers, the clusters obtained in the previous section are needed. Firstly, it is worth mentioning that each cluster is characterized because its members have similar features, at least those ones which have more predictive power for the MLP. Recalling that this similarity is measured by pairwise distances with the Fisher metric, and the Fisher metric is based on the posterior probabilistic model defined by the MLP. Therefore, when segmenting the data into clusters with similar profiles (features), a CI-map can be built per cluster to check whether they have a different feature structure. The expectation is that the most distant clusters present different CI-maps.

From the CI-maps, features that represent the central nodes can be identified as those that are the main drivers for each cluster, looking like a hub in the map. Additionally, to build a Bayesian Network from the CI-map, it is necessary to orient the edges of the CI-map skeleton creating a Directed Acyclic Graph (DAG). The PC-algorithm provides the skeleton and the v-structures of the CI-map, which defines a unique Partially Directed Acyclic Graph (PDAG). There are multiple DAGs compatible with a PDAG, they can be obtained following the edge orienting rules defined in [22, 23], however the order in which the nodes are oriented influences the final DAG. To select one of the multiple DAGs, a predefined node order by mutual information as a base line is proposed, and compared with multiple iterations of random node orders, selecting the DAG with highest BIC score.

The BN provides a probabilistic framework where the probabilities of the model are factorised by the DAG structure, and allows probability estimations conditioned to the evidence of certain nodes. The BN defines some nodes (features) as ancestors or parents, and others as descendants or children, where the probabilities of the children are conditioned to their parents. This enables the creation of Markov blankets to produce additional independences if all the parents of a node are observed as evidence.

In summary, three approaches for data structure identification have been investigated, which use will depend on the chosen class labels. The first, related to the structure of the Fisher manifold created by the classifier model, the second related to the structure

of the data distribution in a Euclidean embedding of the Fisher manifold, identifying the relevant clusters with a characteristic feature profile. And the final approach relates to the structure of feature associations of each cluster in a multivariate correlation model.

1.4 Algorithm pipeline schema

The algorithm pipeline is illustrated in figure 1.1. The pipeline starts with the data (using standardized variables), and a set of class labels which will be used to train the MLP classifier. Steps 2 to 5 belong to the metric learning block, where the outputs are the pairwise distances that form the Fisher manifold, these distances are depicted by an adjacency matrix. Step 6 is used to visualize the Fisher manifold distribution through an Euclidean embedding, observing the distribution will help to inform at the clustering stage whether a data segmentation with spectral clustering or a density-based clustering with the projective methods like PQC is preferred. When the data presents an homogeneous density the data segmentation approach should be followed, the path described in steps 7 and 8 transforms the distances into a similarity network and applies spectral clustering, such as Newman's modularity optimization to find communities in the network. The other path described in steps 9 and 10 embeds the Fisher manifold into a low-dimensional Euclidean space where projective methods can be applied, like histograms if there is a dominant dimension, or data clustering. If the data presents local density variations or non-spherical distributions, the PQC can address this problem quite well.

Although both paths identify the clusters with a set of cluster labels, they are complementary analysing different aspects of data neighbourhood or similarities. The path of spectral clustering (steps 7 and 8) tends to find communities in a similarity network, while the path of clustering based on projective methods (steps 9 and 10) tends to find clusters. Henceforth the terms of communities and clusters will be referred as clusters for simplicity, although they can be defined differently:

- Clusters: groups of similar data identified with projective methods.
- Communities: highly connected networks identified with spectral methods.

Once the relevant clusters are obtained, structure learning methods are applied in steps 11 and 12 to find associations between features, using CI-maps and their derived BNs.

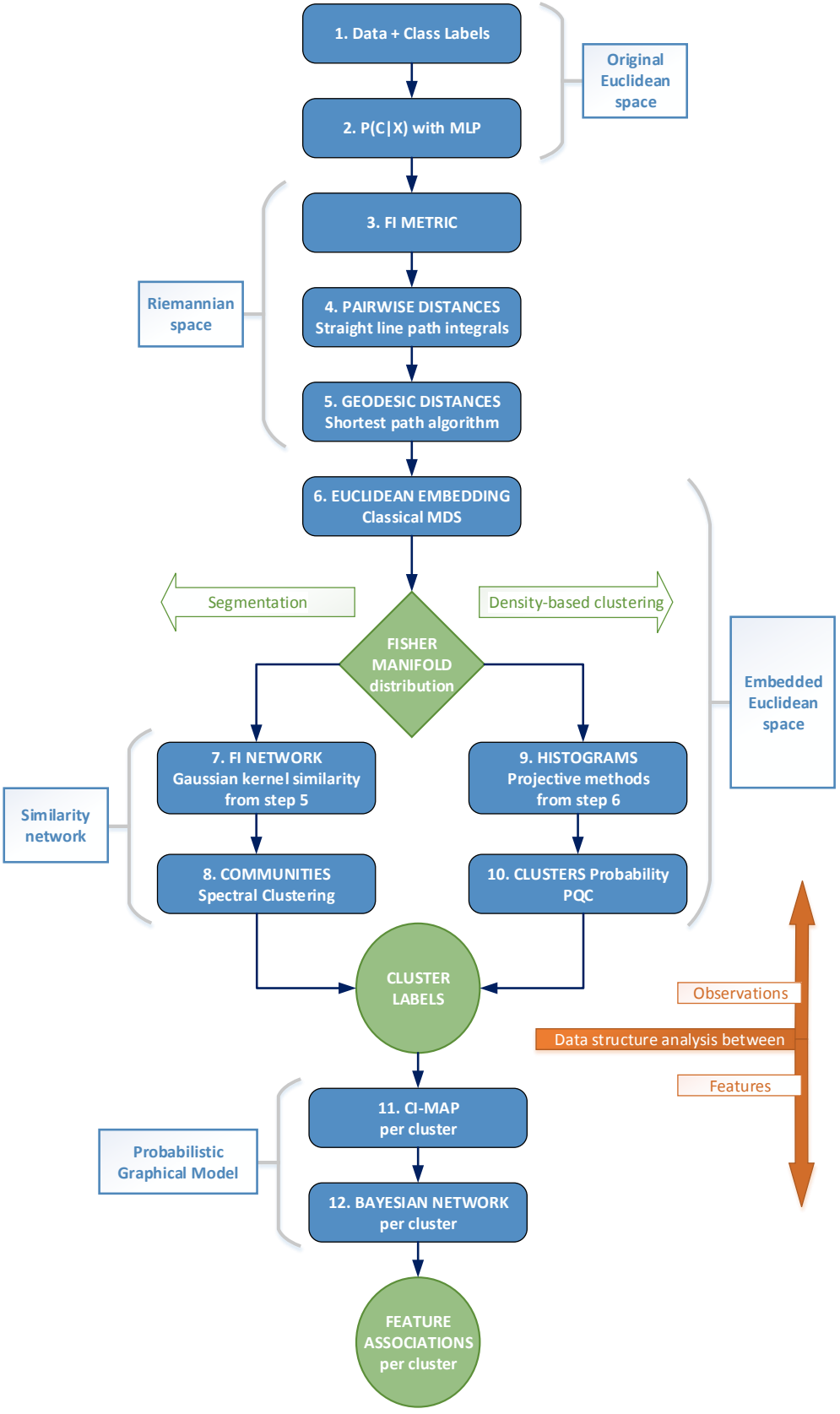


FIG. 1.1 – Algorithm pipeline

1.5 Aims and objectives

The three aims of this work have been motivated from independent sources addressing different problems, united in a common pipeline of methodologies. The main part of the thesis motivation is derived from a collaboration with a big UK retailer company, the company was interested in customer segmentation based on the analysis of shopping baskets. For this task, the methodology of Fisher Information networks fitted quite well with the retail company problem but they were not scalable in a big data environment, this is the origin of the first aim: Develop a scalable implementation of the Fisher Information networks.

The second aim appears when looking for an unsupervised clustering technique that works well in the embedded Fisher manifolds, which tend to be in a low-dimensional space with local density variations. For this task, the method of Quantum Clustering (QC) was proposed because it fits well with the problem. However, the original QC presented a series of limitations that were improved during this work, for instance the problem of selecting the length scale hyper-parameter or the lack of a probabilistic interpretation in the cluster membership.

The third aim appears when feature associations are analysed. The aim is related with the feasibility of building a robust CI-Map from the data, and how a Bayesian network can be derived from the CI-maps.

The list of aims and its objectives following the order of the pipeline methods are presented below:

Aim 1 Develop a scalable implementation of the Fisher Information networks.

- **Obj. 1.1** Evaluate different tools for parallel processing for speeding up the algorithm. These tools must be able to deal with big data environments based on distributed computing.
- **Obj. 1.2** The Multilayer Perceptron (MLP) is the initial candidate for a discriminative classifier, the objective is to check that its scalable implementation achieves analogous results to its non-distributed version in terms of: over-fitting, convergence, neuron weights and performance.
- **Obj. 1.3** Analyse the problem of distance measurement in a Riemannian space, like in the Fisher manifold with a variable metric. The problem is approached from two perspectives: local and global pairwise distances.
- **Obj. 1.4** Find a reasonable approximation of local distances based on path integrals considering the scalability issues.

Obj. 1.5 Find a reasonable approximation of global distances based on shortest path algorithms considering the scalability issues.

Obj. 1.6 Evaluate the runtime bottlenecks of the work flow, in terms of the scalability limitations.

Aim 2 Research clustering techniques suitable to be applied in the Fisher manifolds.

Obj. 2.1 Develop a scalable clustering implementation that links to the pipeline. The initial proposal is based on spectral clustering, since it is the version that was used in the non-distributed implementation.

Obj. 2.2 Test the performance of density-based clustering algorithms, with Quantum Clustering a good initial candidate.

Obj. 2.3 Investigate the hyper-parameter selection in an unsupervised way.

Aim 3 Develop a robust methodology to create Bayesian networks based on CI-Maps from the data.

Obj. 3.1 Analyse the dependency of PC-algorithm policies and hyper-parameters with reference to the skeleton errors of the CI-Maps.

Obj. 3.2 Analyse the sample size influence on the CI-Maps construction

Obj. 3.3 Analyse the problem of node order when the skeleton edges are oriented.

1.6 Contributions

In this thesis three key novelties can be highlighted:

1. **A new method to approximate the All-Pairs Shortest Path (APSP) distance in a Fisher manifold.**

   This method is based on Single-Source Shortest Path (SSSP) Dijkstra's algorithm and it is useful when the APSP Floyd-Warshall algorithm is not available, for instance due to its difficult parallelization in distributed computing. This is explained in Subsection 3.3.2.2 and the algorithm 2 explains the procedure details.

2. **A probabilistic framework for the Quantum Clustering (QC), with two algorithm variants of Probabilistic QC (PQC).**

   Both variants, PQC_{kn} and PQC_{cov}, use a variable local length based on K-Nearest Neighbours (KNN), the first one considers a spherical neighbourhood and the second variant considers non-spherical covariance matrices estimated locally. This is explained in subsections 4.2.2, 4.2.3 and 4.2.4. The algorithm 1 identifies the hyper-parameters of interesting solutions.

3. **A new policy to obtain a explicit Bayesian Network from a CI-Map independent of the node order.**

   The policy orders the nodes by decreasing average mutual information, then orients the edges of the nodes with higher mutual information. Here the nodes represent the features, usually presented as columns in the datasets. This is explained in Subsection 5.2.3 and 5.4.2.

Each key novelty belongs to a different part of the pipeline, and its contribution is integrated in each part as follows:

1.6.1 Scalable FIN implementation

The first contribution is framed under the modification of the FIN algorithms, from step 1 to 8 in the pipeline 1.1, to adapt them in a scalable framework able to deal with a distributed environment and parallel processes.

Part of the FIN methodology is an extension of a previous work [2], already implemented in Matlab but lacking scalability for larger sample size. As explained earlier in section 1.3.1, there are two main bottle necks in the pipeline: the first one is the computation of the $N(N-1)/2$ local pairwise distances with the FI metric, which requires sampling the Fisher information matrix across the path integral (being N the sample size), and the second bottleneck lies in the computation of global distances through shortest path algorithms along the Fisher manifold.

The initial idea was to implement most of the stages of processing in Spark under the Hadoop ecosystem, but several experiments showed that not all the pipeline steps were faster or more reliable in Spark (distributed computing) than they were in Matlab (using a single machine). The runtime of the first bottleneck was considerably improved by the Spark parallelization, nevertheless the second bottleneck was harder to parallelize, requiring new approximations to compute the APSP distances in a distributed environment, here is where the first key novelty is applied. The APSP approximation sacrifices accuracy to obtain better runtime, however, the runtime of the distributed approximation for the APSP (based on Dijkstra's algorithm) is considerably longer than the single machine counterpart (based on Floyd-Warshall algorithm). Therefore, for those cases where the dataset fits into memory of a single machine, a hybrid solution is proposed at the end of section 3.3.3, which consists on implementing the parts that require a higher level of parallelization in Spark, such as Fisher pairwise distances; while performing the other parts in Matlab, such as the shortest path algorithm.

1.6.2 A Probabilistic framework for Quantum Clustering

The second key novelty appears with density-based clustering algorithms in the embedded Fisher manifolds. The original method of Quantum Clustering was a good candidate, however, a potential algorithm improvement was identified; the original version lacked a probabilistic interpretation and a method to select the hyper-parameters in an unsupervised way.

From a theoretical viewpoint, a probabilistic framework for Quantum Clustering that enhances the original algorithm, called PQC, has been developed providing a completely new methodology. Some of its most relevant new features are: local density discrimination, outlier detection, unsupervised assessment of control parameters and insights generation of underlying hierarchical data structure.

The original QC [19] is better able to cluster non-spherical distributions than the K-means algorithm, however QC lacks an assessment method to find an appropriate length scale, which derives the number of clusters, and even when finding an appropriate length scale, the algorithm is not able to fit heteroscedastic data. Hence, there are two challenges that other clustering methods resolve, but not simultaneously. For instance, regular mixtures of Gaussians can deal with heteroscedastic data, but requires a pre-set value of the number of clusters, K. On the other hand, DBSCAN [24] (Density-based spatial clustering of applications with noise) does not need a pre-set of K, but it remains difficult to estimate the hyper-parameter corresponding to the minimum number of core points when the data are heavily heteroscedatic.

However, PQC can resolve these two challenges simultaneously: It has an unsupervised assessment method based on a likelihood function of cluster membership to find the appropriate length scale. Additionally, instead of having a single length scale value, the length scale varies locally according to the local nearest neighbours, capturing the manifold information.

1.6.3 CI-maps stabilization

The third key novelty appears after the CI-Maps are built from the data, when the graph edges are oriented to build a Bayesian network, it happens that different networks are produced depending on the order of the edges being oriented. In fact, the work in this part of the thesis starts with the empirical stabilization of the skeleton errors of the CI-Maps.

From a heuristic and empirical point of view, within the *state-of-the-art* constraint-based structure learning algorithms based on PC-algorithm, the Conditional Independence

map (CI-map) algorithm has been improved where a major challenge was to minimize errors in the graph structure. This work presents empirical evidence for best practice: to reduce false positive errors via the False Discovery Rate (FDR), and to identify appropriate parameter settings to improve the False Negative Reduction (FNR). In addition, several node ordering policies are investigated that transform the skeleton graph into a DAG (edges orienting rules) to form a Bayesian Network (BN). The results show that ordering nodes by strength of mutual information recovers a representative DAG in reasonable time, although a more accurate graph can be recovered using a random order of samples, however this is at the expense of adding significantly to the computation time. The graph accuracy or faithfulness of the BN is measured with likelihood score based on Bayesian Information Criteria (BIC). The purpose of the key novelty is to provide a good baseline score to have a reference for graphs generated by random node order.

1.7 Limitations

This research contains several limitations, some of which could be addressed as future work, other limitations are intrinsic of the problem or require a completely different approach.

1.7.1 FIN runtime limitations

The scalability of the process of measuring distances in a Fisher manifold has runtime limitations with quadratic sample size dependency, $O\left(N^2\right)$. Sample sizes greater than $N > 10^5$ can exceed 24 hours with the Spark implementation used in this work, even with a relatively good computer cluster; specific runtime details about the test performed can be checked in table 3.1 of Chapter 3. As future work, improvements could be obtained changing the parallelizing approach to the GPGPU computing.

1.7.2 QC performance limitations in high-dimensional spaces

The work related with QC has performance limitations related with high-dimensional input data. Fortunately this problem is avoided with the embedded Fisher manifolds as these tend to be low-dimensional. The root of the limitation lies in the measurement of Euclidean distances inside the Gaussian kernels used as a density estimators; the relative distances between observations tend to be very similar in a high-dimensional space, in QC this effect produces the superposition of density estimators based on exponentials with

similar shape that tend to create potential functions with a single potential well that are unable to discriminate the clusters, implying a decrease in performance. This problem is inherent of QC independently of the version of QC, including the new PQC, and it is related with the known *curse of dimensionality* in nearest neighbour search [25, 26].

As future work, this problem could be addressed changing the kernel of the density estimators in such a way that the relative distances can be better discriminated in high-dimensional spaces. However it is not straight forward as there are other QC constraints like the differentiability up to second order of the potential functions. On the other hand, the PQC runtime could be improved using GPGPU parallelization.

1.7.3 Finding true structure from data

One of the main limitations in structure learning is to recover the true structure just from the data without using expert knowledge. In this research, constraint-based algorithms have been used to build the CI-Maps and then the BN, after orienting the CI-Maps skeletons. However, there are currently many families of algorithms to tackle this problem, but none of them can guarantee that the true structure is recovered, both in terms of skeleton errors (if there is an edge or not between two nodes) and the edge orientation of the DAG. Even if the skeleton true structure is found, there are multiple DAGs compatible with each skeleton.

Therefore, assuming the limitation of finding the true structure, the objective is to find the most faithful BN possible, within a sample of compatible BNs, one way to assess a BN is through a likelihood score based on Bayesian Information Criteria (BIC). This work proposes a policy based on node ordering by mutual information to build BN with a reasonably good baseline of BIC score, i.e. a BIC score better than average of those scores obtained from a BN with random node order. This baseline score can be employed as a comparison reference with respect to a new BN created by random order, repeating this process and keeping the BN with best BIC score, guarantees that a reasonably faithful BN is chosen, but not the best or the true one.

1.8 List of papers

This section contains a list of the works published during the research period leading to this thesis, they are listed in chronological order.

- **Performance assessment of quantum clustering in non-spherical data distributions [20].**

Raúl V. Casaña-Eslava, José D. Martín-Guerrero, Ian H. Jarman and Paulo J. G. Lisboa
ESANN proceedings. Bruges, Belgium (2016).

- **Quantum clustering in non-spherical data distributions: finding a suitable number of clusters [21].**
  Raúl V. Casaña-Eslava, Ian H. Jarman, Paulo J. G. Lisboa and José D. Martín-Guerrero
  Neurocomputing (2017).

- Workshop presentation: **Probabilistic Quantum Clustering** Raúl V. Casaña-Eslava
  Quantum Machine Learning & Biomimetic Quantum Technologies. Bilbao, Spain, (March 2018).

- **A Probabilistic framework for Quantum Clustering.**
  Raúl V. Casaña-Eslava, Paulo J. G. Lisboa, Sandra Ortega-Martorell, Ian H. Jarman and José D. Martín-Guerrero.
  arXiv: submit/2577196 [stat.ML] (Feb 2019) [Under revision].

- **Structure finding stabilization and optimization with the PC algorithm.**
  Raúl V. Casaña-Eslava, Ian H. Jarman, Sandra Ortega-Martorell, Paulo J. Lisboa, José D. Martín-Guerrero
  Computational Intelligence methods for Bioinformatics and Biostatistics, CIBB18 (Aug. 2018).

- **Robust CI maps elucidating associations between brain tumours and metabolites observed by MRS.**
  Raúl V. Casaña-Eslava, Sandra Ortega-Martorell, Paulo J. Lisboa, José D. Martín-Guerrero, Ian H. Jarman
  Target journal is Plos One [In preparation].

- **Analysis of data structure with embedded Fisher Information manifolds.**
  Raúl V. Casaña-Eslava, Sandra Ortega-Martorell, Paulo J. Lisboa, José D. Martín-Guerrero, Ian H. Jarman
  Target journal is Neural Computing and Applications [In preparation].

- **Scalable implementation of measuring distances in a Riemannian manifold based on the Fisher Information metric.**
  Raúl V. Casaña-Eslava, Sandra Ortega-Martorell, Paulo J. Lisboa, José D. Martín-Guerrero, Ian H. Jarman
  International Joint Conference on Neural Networks, IJNN (2019).

1.9 Layout of the rest of the thesis

The thesis is presented in seven chapters depicted in figure 1.2, where the main novelty chapters about metric learning, unsupervised clustering and structure learning are highlighted in yellow:

Chapter 1 introduces the overall objectives and the main work-flow of the thesis.

Chapter 2 focuses on the literature review of the three topics: metric learning, unsupervised clustering and structure learning.

Chapter 3, which belongs to the metric learning part, presents a review of the Fisher Information metric, how it can be embedded in a Euclidean space for analysing the data structure with projective methods, and section 3.3 is focused on the scalability of the Fisher Information manifold, applying techniques of distributed computing. Several case studies are presented to illustrate how the algorithm works.

Chapter 4 develops a novel method for unsupervised clustering based on Quantum Clustering in a Euclidean space, this can be applied to the Fisher manifold of previous chapters once the Euclidean embedding is applied. During the chapter small toy examples are used to illustrate how the algorithm works. At the end, several datasets are used for benchmarking the PQC.

Chapter 5 is related to structure learning methods based on Conditional Independence maps and how they can be improved to create more feasible maps and Bayesian networks (BN), just using the information provided by the data. At the end of chapter there is a case study applied to brain tumour dataset.

Chapter 6 presents the whole pipeline applied to two case studies. The main idea is to illustrate the embedded Fisher manifold for these two cases, and to compare both clustering paths on each case. As mentioned before, the option of spectral clustering is good for segmentation, and the option of projective methods with PQC is good for density discrimination. Depending on the manifold distribution and our clustering purposes one path should be chosen, although in this chapter both paths will be compared.

After that comparison, both case studies will be developed in detail. In section 6.3, the music case study analyses the spectral features of the Million Song Dataset [27], its main objective is to compare the preassigned genres with the similarities in the Fisher manifold. In section 6.2, the retail case study applies the pipeline to customer retail data in collaboration with a large UK retailer company (name omitted for confidentiality reasons). The main objective is to perform customer segmentation through shopping basket similarities, later the segmented shopping basket will be used to build BNs, the

structure of each BN can be understood as a shopping basket driver or behaviour, where the more relevant products will be the central nodes.

Chapter 7 summarises the thesis with the final conclusions.

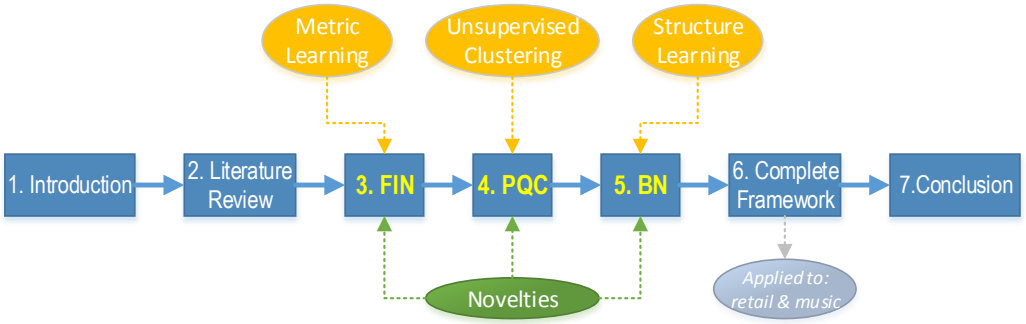


FIG. 1.2 – Schema of the chapters

Chapter 2

Literature review

This chapter presents a literature review about related algorithms employed in each part of the pipeline, the chapter is divided in three main topics: metric learning, clustering and structure learning.

2.1 Metric learning

An important concept in metric learning is the distance function, which intuitively can be seen as a similarity measure between pairs of elements of a data space. This distance function becomes a *metric* if it satisfies the following four conditions: 1) non-negativity, 2) symmetry, 3) triangle inequality and 4) discernibility. If the distance function does not satisfy all conditions it is called *pseudometric*. There are two main categories of metric learning: unsupervised and supervised metric learning.

Unsupervised metric learning is generally associated with methods of dimensionality reduction, where the purpose is to learn a low-dimensional manifold that retains as much information as possible about the geometric relationships between the samples in the original input space under some specific metric. One of the best-known linear data projections is *Principal Component Analysis* (PCA) [28]. Another group of algorithms, not based on linear projections, are the *Multidimensional Scaling* (MDS) [16] algorithms. This family of algorithms starts with a dissimilarity matrix d_{ij} that measures dissimilarities (distances) between observations, and the objective is to find a representation of these observations in an M-dimensional space:
$\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^M$, therefore $d_{ij} \approx ||\mathbf{x}_i - \mathbf{x}_j||$ as close as possible.

From this group of algorithms it is important to highlight the classical MDS [29, 30], also known as *Principal Coordinate Analysis* (PCoA) [31], because this is used for embedding

a Riemannian manifold generated by the Fisher Information metric into a Euclidean space.

From a theoretical standpoint, it is important to mention the Nash embedding theorem [32, 33], which states that every Riemannian manifold of dimension D can be isometrically embedded into some Euclidean space of dimension M, where $M \geq D + 1$. Isometric meaning the length of every path is preserved.

In the case of cMDS, it tries to find a solution $X = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ hence $d_{i,j} = ||\mathbf{x}_i - \mathbf{x}_j||$, where $\mathbf{x}_i \in \mathbb{R}^M$ and $M \geq N - 1$. The solution can be expressed as a function of the $N \times N$ Gram matrix $B = X^T \cdot X$, where now the distances depend on B:

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij} \tag{2.1}$$

where the expression has been obtained taking into account $||\mathbf{x}_i - \mathbf{x}_j||^2 = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i\mathbf{x}_j$, and considering the assumption of centred configuration:

$$\sum_{i=1}^{N} \mathbf{x}_{ik} = 0 \quad \forall \quad k \tag{2.2}$$

This assumption will serve as a constraint for obtaining a unique solution, and for the purpose of dimensionality reduction. Summing for all variables in eq. 2.1, using the constraint eq. 2.2 and rearranging the terms the final solution can be obtained:

$$b_{ij} = \frac{-1}{2\left(d_{ij}^2 - \sum_i d_{ij}^2 - \sum_j d_{ij}^2 + \sum_i \sum_j d_{ij}^2\right)} \tag{2.3}$$

If B is decomposed by its eigenvectors, $B = V \cdot \Lambda \cdot V^T$, then $X = \Lambda^{1/2} \cdot V^T$. In this way X is expressed as the eigenvectors of B, allowing a dimensionality reduction similar to PCA, just discarding the eigenvectors which its eigenvalues have less weight (variance). In fact, the coordinates are ordered from the largest to the smallest variances, allowing any dimension from 1 to M to be selected.

The distance d_{ij} is called a Euclidean distance if exists a finite M therefore:

$$d_{ij} \equiv ||\mathbf{x}_i - \mathbf{x}_j|| \quad \forall \quad i, j \tag{2.4}$$

otherwise d_{ij} is called a non-Euclidean distance, which is the case of the distances obtained in the Fisher manifold.

For Riemannian distances (non-Euclidean) some of the eigenvalues of B are negative, hence these eigenvectors are forced to be discarded. However, in all of the cases where the cMDS with the Fisher manifold has been tested, the M is quite high but the eigenvalues present an exponential decay with a long tail, with the smallest eigenvalues being negative. Therefore they carry little variance, in practice only the first two or three eigenvalues are kept, where the accumulated sum of variance is greater than 80%. On the other hand, when the cMDS is applied to Euclidean pairwise distances the same results as the PCA are recovered, with no eigenvalue being negative.

Apart from the cMDS, special mention should be made of *Sammon mapping* [13] that performs non-linear data projections, and can visualize the input data into a 2D-map using the pairwise distances obtained from the Fisher manifold. Other algorithms are *Isomap* [1] and Local Linear Embedding [34] that assume there is a lower-dimension manifold underlying the original input space, generating a pairwise distance matrix through geodesic lines in the lower-dimension manifold. The algorithms build the path looking for the smallest sample of the k nearest neighbours. It is also worth mentioning the recent non-linear method t-SNE [17] that preserves similarities using a probabilistic formulation with Gaussian kernels and measuring distance distributions with KL-divergence. Minimap [35] is a recent technique that approximates local geodesic distances by shortest paths along a neighbourhood graph adding a penalizing factor depending on the number of steps in the path, combined with Sammon mapping it enhances the local structures at the expense of long distances.

Supervised metric learning uses class labels to define similarity patterns in the input data. Nearest neighbour classifiers [36, 37] compute the output function based on the values of the known output functions in the surrounding area. Other methods like *large margin nearest neighbour* (LMNN) [38] learns a Mahalanobis metric through the cost function minimization that tries to pull together neighbours of the same class and to push out neighbours of different classes. Similar works on Mahalanobis metric are *neighbourhood component analysis* (NCA) [39] and *global similarity distance metric* (GSDM) [40].

There is another family of metric learning inspired by *Fisher's linear discriminant analysis* (LDA) [41] where a linear transformation searches for the maximum separation between the classes (assuming the data is normally distributed). In these algorithms the between-class and within-class covariance matrices play an important role in the metric tensor. Two such examples are *discriminant adaptive nearest neighbour* (DANN) [42] and *relevant component analysis* (RCA) [43, 44].

Another case of metric learning are similarity measures by means of kernel functions. Usually kernel functions map the input space into a higher dimensional space, and since kernel functions can be arbitrarily complex, they can be designed in such a way

that linear relationships in the kernel space, like *inner product*, represent non-linear relationships in the input space, this is known as the *kernel trick*:

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right) \cdot \phi\left(\mathbf{x}_j\right) \tag{2.5}$$

Kernel similarities can be interpreted as distance measures of the input space images in the kernel manifold. In other words, there is a mapping between the points of the input space and their images in the manifold, where the distances between images can be measured:

$$d_\phi\left(\mathbf{x}_i,\, \mathbf{x}_j\right) = d_{\mathcal{M}}\left(\phi\left(\mathbf{x}_i\right), \phi\left(\mathbf{x}_j\right)\right) \tag{2.6}$$

The kernel manifold $\mathcal{M}$ usually has a curved surface, creating a *Riemannian manifold*, where the distances are estimated locally using the *Riemannian metric tensor* [45] $\mathbf{G}\left(\mathbf{x}\right)$.

$$d_\phi\left(\mathbf{x}, \mathbf{x} + d\mathbf{x}\right)^2 = d\mathbf{x}^T \cdot \mathbf{G}\left(\mathbf{x}\right) \cdot d\mathbf{x} \tag{2.7}$$

$\mathbf{G}\left(\mathbf{x}\right)$ is a positive definite matrix that can be expressed in terms of infinitesimal displacements in the kernel manifold. Let $\mathbf{z} = \phi\left(\mathbf{x}\right)$ and $\mathbf{z} + d\mathbf{z} = \phi\left(\mathbf{x} + d\mathbf{x}\right)$. Then you have:

$$d\mathbf{z} = \phi\left(\mathbf{x} + d\mathbf{x}\right) - \phi\left(\mathbf{x}\right) \approx \nabla_{\mathbf{x}} \phi\left(\mathbf{x}\right) d\mathbf{x} \tag{2.8}$$

$$d_\phi\left(\mathbf{x},\, \mathbf{x} + d\mathbf{x}\right)^2 = \|d\mathbf{z}\|^2 = d\mathbf{x}^T \cdot \nabla_{\mathbf{x}} \phi\left(\mathbf{x}\right)^T \nabla_{\mathbf{x}} \phi\left(\mathbf{x}\right) \cdot d\mathbf{x} \tag{2.9}$$

Approximating $\phi\left(\mathbf{x} + d\mathbf{x}\right)$ up to first order Taylor expansion. Then, using the kernel trick function: $K\left(\mathbf{x}, \mathbf{y}\right) = \phi\left(\mathbf{x}\right) \cdot \phi\left(\mathbf{y}\right)$, the equation 2.7 can be expressed as:

$$d_\phi\left(\mathbf{x},\, \mathbf{x} + d\mathbf{x}\right) = d\mathbf{x}^T \cdot \nabla_{\mathbf{x}} \nabla_{\mathbf{y}} K\left(\mathbf{x}, \mathbf{y}\right)\big|_{\mathbf{y}=\mathbf{x}} \cdot d\mathbf{x} \tag{2.10}$$

$$\mathbf{G}\left(\mathbf{x}\right) = \nabla_{\mathbf{x}} \nabla_{\mathbf{y}} K\left(\mathbf{x}, \mathbf{y}\right)\big|_{\mathbf{y}=\mathbf{x}} \tag{2.11}$$

So, there is a strong relationship between kernel functions and the tensor metric.

For estimating distances between non-adjacent points in $\mathcal{M}$ the following path integral is needed:

$$d_\phi\left(\mathbf{x}_i,\, \mathbf{x}_j\right) = \left| \int_{t_i}^{t_j} \sqrt{\dot{\mathbf{x}}(t)^T \mathbf{G}\left(\mathbf{x}(t)\right) \dot{\mathbf{x}}(t)} dt \right| \tag{2.12}$$

where $t \in [t_i, t_j]$ parametrizes the path in $\mathbf{x}(t)$, from $\mathbf{x}_i = \mathbf{x}(t_i)$ to $\mathbf{x}_j = \mathbf{x}(t_j)$, and $\dot{\mathbf{x}}(t) = d\mathbf{x}/dt$.

Only when the manifold is flat, i.e. lies in a hyperplane and $\mathbf{G}\left(\mathbf{x}(t)\right)$ is constant, the global pairwise distances can be computed using straight lines connecting the points, like in Euclidean space. Otherwise, the distances have to be computed locally and $\mathbf{G}\left(\mathbf{x}(t)\right)$ needs to be evaluated across the path.

One kernel of interest for this thesis is the *Fisher kernel* [46], which measures the similarity of two points with reference to a generative statistical model $p(\mathbf{x}|\theta)$, parametrized by θ. This generative model produces a manifold where the *Riemannian metric tensor* [47] is the *Fisher information* matrix [48]:

$$d\left(\theta,\, \theta + d\theta\right)^2 = d\theta^T \mathbf{G}\left(\theta\right) d\theta \tag{2.13}$$

$$\mathbf{G}\left(\theta\right) = E_x\left[\nabla_\theta \log p(\mathbf{x}|\theta) \cdot \nabla_\theta \log p(\mathbf{x}|\theta)^T\right] \tag{2.14}$$

where E_x is the expected value with reference to $p(\mathbf{x}|\theta)$.

With the Fisher metric, distances between two close parameters θ and $\theta + d\theta$ correspond to manifold distances between the corresponding densities $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\theta + d\theta)$ in terms of the expected variation in the log-likelihood of $\mathbf{x}$. However, because the metric is Riemannian and not Euclidean, the expected variation of $\log p(\mathbf{x}|\theta)$ for the same $d\theta$ distortion depends on the location of the space $\mathbf{x}$, in which the log-likelihood variation is measured.

The Fisher kernel of two points in the input space, given a generative model $p(\mathbf{x}|\theta)$, is defined as:

$$\begin{aligned} K\left(\mathbf{x}_i, \mathbf{x}_j\right) &= \mathbf{n}\left(\theta, \mathbf{x}_i\right)^T \cdot \mathbf{G}\left(\theta\right) \cdot \mathbf{n}\left(\theta, \mathbf{x}_j\right) \\ &= \nabla_\theta \log p(\mathbf{x}|\theta)^T \cdot \mathbf{G}\left(\theta\right)^{-1} \cdot \nabla_\theta \log p(\mathbf{x}|\theta) \end{aligned} \tag{2.15}$$

where the *natural gradient* [49] has been used to find the θ direction of the steepest ascent of the log-likelihood at point $p(\mathbf{x}|\theta)$ of the manifold:

$$\mathbf{n}(\theta, \mathbf{x}) = \mathbf{G}(\theta)^{-1} \nabla_\theta \log p(\mathbf{x}|\theta) \tag{2.16}$$

Focusing on the FI metric, it is worth mentioning its connection with the KL divergence [9], measuring the difference between adjacent probability distributions on the manifold:

$$\begin{aligned} I_{KL}(\theta : \theta + d\theta) &= I_{KL}(p(\mathbf{x}|\theta), p(\mathbf{x}|\theta + d\theta)) = d\theta^T \mathbf{G}(\theta)\, d\theta \\ &= -\int \log\left(\frac{p(\mathbf{x}|\theta + d\theta)}{p(\mathbf{x}|\theta)}\right) p(\mathbf{x}|\theta) d\mathbf{x} \end{aligned} \tag{2.17}$$

Generally, most of the works in the literature that involve the FI metric [9, 45, 46, 48–50] use the metric defined in parameters manifold based on generative models, $p(\mathbf{x}|\theta)$. However based on [51, 52], it is possible change the approach and apply the Fisher metric on discriminative models $p(y|\mathbf{x})$ that classify an external auxiliary information y. In this case, the metric measures parameter distortions with reference to the input space $\mathbf{x}$ instead of θ.

This is the main idea followed in chapter 3 to build the Fisher manifolds based on the new approach of the Fisher metric.

On the other hand, one of the tasks derived from the computation of distances in this Riemannian space is how to estimate global distances in a space where the metric changes locally. This task cannot be addressed analytically with an exact solution because is infeasible to compute the metric of all possible paths between two points to check which path is shorter, therefore several approximations have to be taken. The first approximation is to compute local distances with path integrals in a similar way described in eq. 2.12 but sampling the metric in regular intervals between the two points. The other approximation tackled in this thesis is to use these local distances to build a fully connected network (graph) of distances, where the nodes are observations and the edges their pairwise local distances. With this network, the global distances can be approximated computing the shortest path between nodes. Two algorithms have been used to perform the shortest path between nodes in a graph: the Floyd-Warshall algorithm [53, 54], which belongs to the class All-Pairs Shortest Path, and the Dijkstra's algorithm [55], which belongs to the class Single-Source Shortest Path. The APSP class computes the shortest distances of all nodes respect to all nodes, the SSSP class computes the shortest distance of one node respect to all nodes. The APSP class fits better to the problem of this thesis, but it is not always available like in a distributed computing environment.

2.2 Clustering

The second part of the pipeline is focused on clustering the Fisher manifold, where the Fisher manifold is initially described by an adjacency matrix with the pairwise distances, and then, with the information of the adjacency matrix, can be embedded in a Euclidean space to visualize the manifold distribution. These two possible representations of the Fisher manifold offer at least two options to tackle the clustering problem.

The first option is to work directly with the adjacency matrix through spectral clustering methods. This requires a transformation of the distance adjacency matrix into a similarity network, where the range of values of the matrix elements are transformed from distances $\in [0, \infty[$ to similarity scores $\in [1, 0]$.

Thus, spectral clustering methods usually work with similarity networks for finding communities, and the clustering task is usually called community detection. The outcome is a set of labels identifying each community. However in terms of notation, as mentioned before in section 1.4, communities will be referred as clusters for simplicity, although communities can be defined as highly connected networks identified with spectral methods. The next section 2.2.1 reviews some of the spectral clustering methods.

The second option is to work with the Fisher manifold embedded in a Euclidean space, in such a way that projective methods can be applied. The main difference is that now each observation is defined by a coordinates instead of pairwise distances. In this new Euclidean space the standard clustering methods described in section 2.2.2 can be implemented.

One of the advantages of spectral clustering is that it can work directly with adjacency matrix, i.e. an Euclidean embedding is not needed, and spectral clustering scales better with the sample size. The disadvantage is that an additional hyper-parameter is required to set the length scale of the similarity network. Alternatively, projective methods in a Euclidean embedded space have the advantage of being able to use more sophisticated algorithms, like those based on density discrimination.

2.2.1 Community detection with spectral methods

The starting point is a similarity network (graph) where the nodes represent observations and edges represent pairwise relations, this network has been obtained after applying a similarity kernel on the FI adjacency matrix.

Intuitively, the idea of community is understood as a group of nodes densely interconnected, the number of edges between groups is much smaller in comparison with the intra-group edge density [56–58].

A similarity network can be obtained from the pairwise distances of the Fisher manifold using a Gaussian radial kernel that transforms distances into similarities:

$$A_{ij} = \exp\left(-\frac{dist\left(\mathbf{x}_i, \mathbf{x}_j\right)^2}{\sigma_G^2}\right) \tag{2.18}$$

where σ_G^2 is the length scale that controls the kernel width (community size), the smaller the length scale the larger the number of communities. This length scale can be estimated empirically measuring the faithfulness of the network predictions compared with the classifier model, or can be estimated heuristically based on a percentage of the average Fisher distances between the samples that belong to the same class label, the percentage will define the grade of granularity of the network.

Regarding community extraction algorithms, there are multiple options in the literature. There are, to mention a few, graph partitioning algorithms, which are based on graph iterative bisections, for instance the Kernighan-Lin algorithm [59], or spectral partitioning [60] based on the spectral properties of the Laplacian matrix to perform the bisection. Other approaches are hierarchical clustering [61] with agglomerative methods, divisive hierarchical methods like the Girvan-Newman algorithm [62], which is focused on removing edges between modules instead of removing edges with low similarity. Here the concept of modularity [63] is important, which is a measure of the quality of the division of a network. Some of the most relevant modularity optimisation algorithms are [64], the greedy optimization method based on agglomerative hierarchical clustering, [65] uses a simulated annealing, [66] maximizes the modularity through an extremal optimisation, and [12] is based on modularity spectral optimisation. The last one will be used in the pipeline with a Matlab implementation.

The modularity spectral optimisation method will be used as a default clustering method during the Chapter 3, where the Fisher Information networks and its scalable version will be presented. Only for visualization purposes, an Euclidean embedding of the Fisher manifold will be performed in this chapter.

2.2.2 Clustering with projective methods

The starting point is an embedded Fisher manifold in a low-dimensional Euclidean space. In this work the cMDS is the preferred algorithm for the embedding task, because it

preserves the global distances and the dimensionality is adapted to the Fisher manifold complexity (the more complex the more dimensions needed for the embedding).

In the literature there are many clustering algorithms, they can be classified as: centroid-based clustering like the well known K-Means [67], connectivity-based clustering (hierarchical) [68], distribution-based clustering like Gaussian mixture models [61], and density-based clustering like DBSCAN (Density-based spatial clustering of applications with noise) [24] or the less known Quantum Clustering [69].

In this section there is more interest in density-based clustering because the Fisher manifold distributions tend to have local density variations. However, other kind of algorithms also have been tested, for instance K-means obtains similar results to the ones obtained with spectral clustering, at least in the examples performed in this work where the manifold distributions had an homogeneous and the problem was a segmentation task. Conversely, when the manifold has strong density variations, neither the K-means nor the spectral clustering are able to discriminate properly the density clusters.

From the density-based clustering algorithms, the original QC was the initial proposal due to a work [21] about unsupervised length scale selection based on concordance measures showed promising results. The use of concordance measures to select the hyper-parameter was previously employed with K-means in [18, 70], where the Separation-Concordance (SeCo) framework was used to select the number of K. The concordance, which is based on Cramer's V statistic [71], measures the similarity among different cluster solutions with stochastic algorithm initializations.

A further research on how to improve the original QC with its fixed length scale led to a new QC reformulation under a probabilistic framework. The new probabilistic QC improves the original one, with new characteristics that fit very well for the clustering task in the Fisher manifold. The algorithm highlights are introduced in the next subsection.

2.2.2.1 Probabilistic Quantum Clustering

Quantum Clustering (QC) is an appealing paradigm inspired by the Schrödinger equation [69] with potential to identify and track connected regions while separating them from other nearby clusters. However, the method would benefit from a stronger theoretical basis to assess the goodness of fit to the density distribution of the data, in particular to guide the choice of control parameters which determine the number of clusters detected. In addition, QC is prone to fragment the data into many small clusters that may comprise outliers, without clearly defined means to control this process.

A probabilistic framework to address these issues is proposed. This framework is applied to an extension to QC using local estimates of the central length scale parameter, which enables the model to accurately detect clusters with very different data densities.

The starting point is the original quantum clustering algorithm introduced by [69] which generates a potential function $V(\mathbf{x})$ from a wave function $\Psi(\mathbf{x})$ as a constant energy solution of the time-independent Schrödinger equation:

$$H\Psi \equiv \left(-\frac{\sigma^2}{2}\nabla^2 + V(\mathbf{x})\right)\Psi(\mathbf{x}) = E\Psi(\mathbf{x}) \tag{2.19}$$

where H is the Hamiltonian and E the constant total energy.

In the original formulation the wave function represented a Parzen estimator with a given length scale parameter, σ. Given a potential function generated from the wave function using the Schrödinger equation, the allocation of individual data points to clusters was determined by the use of gradient descent [69] to find local minima and allocate clusters based on maximum probability of cluster membership, although local Hessian modes in a potential lattice have also been used for this purpose [72].

While the wave function providing data density estimates need not be Gaussian e.g., B-splines [73], Vector Quantization [74] or the Epanechnikov kernel [75, 76] with optimal efficiency, exponential distributions are generally preferred due to their smoothness since the wave function has to be differentiable up to third order in the potential function:

$$V(\mathbf{x}) = E + \frac{\sigma^2}{2}\frac{\nabla^2\Psi(\mathbf{x})}{\Psi(\mathbf{x})} \tag{2.20}$$

To define the wave function is proposed by assigning a normalized Gaussian function to each observation in the training set, centred on the observed data point and with the covariance matrix estimated locally from a set number of nearest neighbours (NNs) and assumed to be diagonal.

This is in contrast with Mixtures of Gaussian Models (MGMs) where the mean and covariance parameters are not tied to data points and are fitted using Maximization-Expectation [61]. The new approach is equivalent to a generative model with a kernel representing inherent noise and a prior that is uniformly distributed across all of the observations. The total wave function thus comprises many narrow Gaussians creating an aggregate density function that links neighbouring data points to generate a smooth and connected valley in the potential function.

Clearly the length scale of the exponential functions, σ, which parameterises the variance of the noise estimate, is of critical importance since it determines the overlap between the wave function components from neighbouring observations and so has a critical impact on the shape and smoothness of the resulting potential function, by affecting the number of local minima and, consequently, also the number of clusters.

A recent publication [77] recognises the difficulty in mapping local data structure with local potential functions and proposes training a self-organised neural network with radial basis functions as the basic computational units. This empirical approach is effective although it does not claim to optimise the model parameters, because of the complex structure of local minima of the potential surface.

Regarding metric learning,Topological Data Analysis (TDA) [78, 79] can be considered a similar approach in terms of data-structure characterization through the search of distance-parameter stabilization, where the clustering is based on this topology persistence [80]. However, TDA lacks the methodical and objective criteria for selecting the hierarchical level of the dendrograms. On the contrary, we propose a Bayesian interpretation of our generative model, in order to infer probabilistic measures for the goodness-of-fit of the data, providing a score function for parameter selection. The probabilistic QC (PQC) outperforms TDA in terms of Jaccard scores (JS), as it will be shown in Section 4.4.

It was the dependence of the original QC on the band-width selection of the Parzen window which originally led to the use of k-nearest neighbours (KNN) in kernel estimators of the local sample covariance [72, 81]. Unfortunately, the efficiency of KNN estimators varies considerably depending on the structure of the data [21]. An alternative approach is considered in [82, 83], where the kernel scale is locally estimated. In [84], a probability density function is estimated using a manifold Parzen window, rendering the Gaussian function non-spherical. Summing up, the determination of a suitable kernel length to discriminate clusters from the QC potential lacked a defined framework to measure goodness of fit to the data, making it difficult to optimise this critical parameter.

A probabilistic interpretation of quantum clustering is proposed through the use of wave functions comprising normalised joint probability distributions. This enables the length parameters for local covariance estimation to be optimised by maximising a Bayesian probability of cluster allocation. An empirical evaluation with synthetic and real-world data shows that the approach is robust for clustering complex data by maximising the probability of cluster membership without prior knowledge of the correct number of clusters. After obtaining the probability of cluster membership, the standard Bayesian framework can be used to detect outliers.

Furthermore, a positive definite likelihood function of cluster membership can be optimised to select the bandwidth of the Gaussian functions, namely the number of KNNs used for local covariance estimation, which is the only free parameter in the model. This underlines PQC as a plausible method for the detection of hierarchical data structure.

The proposed framework addresses two complementary challenges for current methods. Regular mixtures of Gaussians cannot resolve the number of clusters, hence requiring a preset value of K. While this is addressed in part by DBSCAN [24], it still remains difficult to estimate the hyper-parameter corresponding to the minimum number of core points when the data are heavily heteroscedastic. Therefore, this part of the thesis shows a framework to select efficient parameters to map, with quantum clustering, complex data structure with no prior knowledge of the number of clusters.

2.3 Structure learning

In this work, the structure learning methods are applied on the data stratified by clusters with the purpose of obtaining more defined feature relationships than would have been achieved considering all the data. In any case, the same methods described below could be applied to the entire data set.

Structure finding methods lie within the field of Probabilistic Graphical Models and are extensively studied [85, 86], especially from a theoretical perspective, as they offer an efficient graphical approach to apply statistical estimates in a complex system. They serve as a framework for Bayesian and Markov networks [7], and have two components: a structure in the form of a graph, and a set of parameters that can be used to make statistical estimations.

Recently, a new structure finding algorithm [87] was proposed, which can obtain a faithful Bayesian Network (BN) without the need for specific approximations and with a reasonable computation time. Using these methods, complex associative maps can be obtained through Conditional Independence Maps (CI-maps). Many constraint-based structure learning algorithms, including this work, are based on the PC-algorithm [8], where the graph starts fully connected graph, and edges are removed between nodes (variables) based on pairwise independence tests, increasing the number of conditioned variables as the algorithm progresses. The algorithm stops when it finally converges to a stable structure forming a CI-map. This work extends the methods used in [87], where the criterion for independence is based on conditional mutual information instead of likelihood-ratio tests (G-test). In addition, this work proposes several policies to create a data-driven structure:

- FDR [88] controls the False Positives decreasing the significance level in conditional independence tests when they are applied multiple times on the same nodes. This translates to reduced number of edges.
- FNR [89] tries to avoid independence tests if it is not powerful enough. The criteria is based on a threshold of Degrees of Freedom (DoF) that depends on the desired power for the test, the sample size and the effect size.
- The Weakest First (TWF) affects the order in which the graph is being pruned. The outcome of the PC algorithm is influenced by the order in which the conditional independence tests are executed. TWF sorts the nodes by mutual information (edge strength), testing first the weakest nodes which reduces the problem of incorrect pruning or incorrect edge dependence discovery.

Once the structure is found, the next step is to build a DAG following the rules defined in [22]. These rules do not necessarily lead to a unique DAG, where the most general solution is a Partial Directed Acyclic Graph (PDAG).

It is well known that the PC algorithm is sensitive to the order in which the nodes are tested [90, 91]. This problem is addressed by using a similar solution to TWF, but ordering the nodes by descending mutual information. This policy is called The Strongest First (TSF). The DAG obtained with TSF node order is compared with DAGs generated by random node order, the BNs derived from the DAGs are assessed by the log-likelihood function with the Bayesian Information Criteria (BIC), which penalizes the graph complexity.

Other studies take different approaches, such as the Complete Partially Directed Acyclic Graphs used in [92], latent variables with the Fast Causal Inference algorithm in [8], Maximal Ancestral Graphs in [93], IDA algorithm (Intervention calculus when the DAG is Absent) in [94, 95], and Essential Graph Search in [96] based on scoring random order DAGs.

2.4 Conclusion

In this chapter, many related algorithms from completely different areas of machine learning have been reviewed. The part of metric learning contains the foundations of the Fisher manifold, which is the main driver of the thesis. The modification of the Fisher metric to work with the input space is an algorithm quite specific, and its choice has been motivated by its potential application in the analysis of shopping baskets.

However, in the clustering part of the pipeline, several algorithms could have been used to cluster the manifold. The Newman's algorithm based on spectral clustering was chosen for the ability to work directly with the pairwise distance matrix and for its scalability. On the other hand, the new PQC was chosen for the ability to discriminate clusters by density and for being able to select its hyper-parameters in an unsupervised way.

Analogously, during this chapter several algorithms capable of doing the structure learning task have been mentioned, the reason why the PC-algorithm has been chosen is because of its ability to customize hyper-parameters and policies, and because of the low error in the structures (skeletons) created from the data.

Nevertheless, this work does not claim that the chosen algorithms are the best or the only algorithms capable of doing the pipeline tasks.

The next three chapters (3, 4, 5) will introduce the details of the methodologies of the Fisher manifold, PQC and CI-maps respectively.

Chapter 3

Fisher Information Networks

The chapter is divided in two blocks, the first block introduces the theoretical framework to build a Fisher manifold and describes the methodology used when data sample size is small enough to avoid runtime problems. The second block introduces a scalable implementation that tries to mitigate the runtime problems that appear when the size of the data grows.

The first part starts with the foundations of the Fisher manifold. Then the methodology section introduces the concept of Fisher Information matrix interpreted as a metric. An important adaptation is that the Fisher metric is derived from a discriminative classification model, defining a metric based on the input space using a MLP as the discriminative model. This is followed by the methodology needed for estimating the global pairwise distances in the Fisher manifold. The next section covers the manifold transformation into a similarity network, and how to determine the length scale of the Gaussian kernel to obtain a reasonable number of communities through spectral clustering algorithms. This is followed by an introduction to low-dimension visualization methods for representing the FIN.

The second part focuses on a scalable implementation based on distributed computing with Spark, the section analyses the bottlenecks of the pipeline, where two approximations are proposed to tackle the shortest path task in a distributed environment, and then the runtime of different configurations are studied.

The chapter ends with several case studies: The first one is based on aneurysm data and it is the most extensive, for this case the data is small and can be addressed avoiding the scalable implementation. The aim is to find clusters within the Fisher manifold that are able to classify patients presenting a ruptured aneurysm. The other two case studies use the scalable implementation, one is a synthetic non/linear dataset based on nested

spirals, the other is data from Monte Carlo simulations of high-energy physics particle detection.

Finally, this chapter is brought to a close with the conclusions.

3.1 Foundations of Fisher Information manifold

3.1.1 Fisher metric in the input space

Generally, most of the works in the literature that involve the FI metric [9, 45, 46, 48–50] use the metric defined in parameters manifold based on generative models, $p(\mathbf{x}|\theta)$. However based on [51, 52], it is possible change the approach and apply the Fisher metric on discriminative models $p(y|\mathbf{x})$ that classify an external auxiliary information y. In this case, the metric measures parameter distortions with reference to the input space $\mathbf{x}$ instead of θ.

$$d\left(\mathbf{x},\, \mathbf{x}+d\mathbf{x}\right)^2 = d\mathbf{x}^T \mathbf{G}\left(\mathbf{x}\right) d\mathbf{x} \tag{3.1}$$

$$\mathbf{G}\left(\mathbf{x}\right) = E_y\left[\nabla_{\mathbf{x}} \log p(y|\mathbf{x}) \cdot \nabla_{\mathbf{x}} \log p(y|\mathbf{x})^T\right] \tag{3.2}$$

Generally, the auxiliary information y represents a class label C composed by J discrete values, $y \in [c_1, \cdots, c_J]$. The probability function $p(y|\mathbf{x})$ represents the discrete probability distribution conditioned on $\mathbf{x}$, where the Expected value E_y over $p(y|\mathbf{x})$ is the summation of FI matrix over each class in $p(y|\mathbf{x})$:

$$\mathbf{G}\left(\mathbf{x}\right) = \sum_{j=1}^{J} \nabla_{\mathbf{x}} \log p(c_j|\mathbf{x}) \cdot \nabla_{\mathbf{x}} \log p(c_j|\mathbf{x})^T \, p(c_j|\mathbf{x}) \tag{3.3}$$

This metric measures local distances in the input space $d\mathbf{x}$ as a function of the variations on the class probabilities, assigning longer distances in the direction of the posterior probabilities that have more variation, and shorter distances where there are no class probability changes. In other words, the FI metric contains local relevant information about the probability rate of change of class y membership.

The new Fisher metric approach still has a connection with the KL-divergence:

$$
\begin{aligned}
I_{KL}\left(\mathbf{x}:\mathbf{x}+d\mathbf{x}\right) &= I_{KL}\left(p\left(y|\mathbf{x}\right),p\left(y|\mathbf{x}+d\mathbf{x}\right)\right) \\
&= -\sum_{j}^{J}\log\left(\frac{p(c_j|\mathbf{x}+d\mathbf{x})}{p(c_j|\mathbf{x})}\right)p(c_j|\mathbf{x}) \\
&= -\sum_{j}^{J}\left(d\mathbf{x}^T\nabla_{\mathbf{x}}\log p(c_j|\mathbf{x})+\frac{1}{2}d\mathbf{x}^T\left(\nabla^2_{\mathbf{x}}\log p(c_j|\mathbf{x})\right)d\mathbf{x}\right)p(c_j|\mathbf{x}) \\
&= -\frac{1}{2}d\mathbf{x}^T\left(\sum_{j}^{J}\left(\nabla^2_{\mathbf{x}}\log p(c_j|\mathbf{x})\right)p(c_j|\mathbf{x})\right)d\mathbf{x} \\
&= \frac{1}{2}d\mathbf{x}^T\left(\sum_{j=1}^{J}\nabla_{\mathbf{x}}\log p(c_j|\mathbf{x})\cdot\nabla_{\mathbf{x}}\log p(c_j|\mathbf{x})^T\,p(c_j|\mathbf{x})\right)d\mathbf{x} \\
&= \frac{1}{2}d\mathbf{x}^T\cdot\mathbf{G}(\mathbf{x})\cdot d\mathbf{x}
\end{aligned}
\tag{3.4}
$$

where $\log p(c_j|\mathbf{x}+d\mathbf{x})$ has been approximated with the second order of Taylor expansion, and some terms are cancelled due to $\sum_j^J p(c_j|\mathbf{x})=1$.

3.1.2 Global distances in Fisher manifold

There is a problem to compute optimally global pairwise distances. Global distances are computed following a similar path integral shown in eq. 2.12, but the optimal path where $\mathbf{G}\left(\mathbf{x}\right)$ is evaluated is unknown, and it is not practical to evaluate $\mathbf{G}\left(\mathbf{x}\right)$ everywhere. Therefore, in practice approximations are necessary and are described in more detail in the methodology section 3.2.2. The first group of assumptions is called the *straight line approach*, and it considers straight line paths between points, where $\mathbf{G}\left(\mathbf{x}\right)$ is sampled in regular intervals across the straight line, the integral path is approximated as a sum of small segments where each segment is considered with a constant metric.

The second assumption, called the *shortest path approach*, improves the pairwise distances computed with the *straight line approach*. The idea is to use shortest path algorithms, like the Floyd-Warshall algorithm [53, 54] or the Dijkstra's algorithm [55], to find a shortest path in the manifold of pairwise distances generated by the *straight line approach* to make the global distances shorter. The Floyd-Warshall algorithm, which belongs to the family *All Pairs Shortest Path* (APSP), will be implemented in Matlab (single machine) and will be used in the procedure of this chapter. The Dijkstra's algorithm, which belongs to the family *Single Source Shortest Path* (SSSP), will be used a modified version in Spark (distributed computing) to apply sequentially the algorithm

over certain sources (nodes) of the manifold to approximate an APSP using an SSSP algorithm, it will be used in the pipeline of this chapter, 3.

After the shortest path process, the pairwise distances under the Fisher manifold are considered as faithful as possible within a reasonable runtime, due to the runtime at least depends on $O\left(N^2\right)$, with N being the sample size.

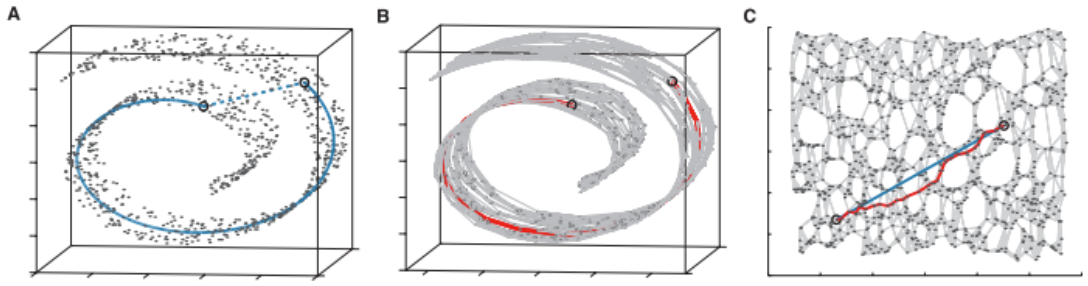


FIG. 3.1 – Manifold example with Isomap from [1]

As an example, figure 3.1 shows a "Swiss roll" manifold and how Isomap reduces the features dimensions. The image is taken from the original paper by Tenenbaum et al. [1]. Fig. 3.1-A shows two points relatively close in the Euclidean space, but far away within the manifold created by the "Swiss roll". Fig. 3.1-B shows the graph generated by K-NN with K=7 on the "Swiss roll" manifold, and the shortest path applying Djikstra's algorithm. Fig. 3.1-C shows the 2D manifold representation generated with cMDS using the matrix of graph distances, additionally the geodesic path of the two points is compared with the Djikstra's shortest path.

3.2 Methodology

This section describes the methods applied in the research for small sample sizes. The algorithms are implemented in Matlab using a single machine (non-distributed computing) via CPU. The stages of processing include the MLP classifier, Fisher metric, pairwise distances in the Fisher manifold, transformation into similarity networks for community finding, and Euclidean embedding for visualizing the structure of the Fisher manifold with the communities.

It is important to mention that small sample sizes are considered to be lower than 5000 observations. Although the pipeline runtime also depends on the number of features and the number of class labels, the decisive factor is the sample size due to the Fisher pairwise distance computation, which, as mentioned before, is the main pipeline bottleneck. For instance, in a standard computer with I7 CPU and 8GB RAM, the runtime is less than 1 hour for sample sizes lower than 1000 observations. However, for 7500 observations

the runtime can roughly take 10 days, this is the main reason why a scalable version of the FIN has been implemented in the next chapter.

3.2.1 Fisher metric

Let us suppose there is a dataset with standardized numerical features, if they are categorical data One-Hot Encoding can be used to make dummy variables, and also there is a set of class categorical labels which will be used as a target variable for the classifier model.

The starting point is the selection of the classifier model. This will need to a) be a multinomial classifier, b) be able to deal with non-linear data, c) have a mechanism to avoid overfitting, and d) allow the model to be easily differentiable up to second order with respect to the input space. The latter requirement is due to the core of the manifold being the Fisher Information matrix [48], which can be derived as the Hessian of the KL-divergence (also called relative entropy), implying derivatives of second order. Given these requirements, a MLP classifier regularized with weight decay was chosen for the task.

Using the MLP as a discriminative model, the posterior probability estimation is evaluated with the soft-max activation:

$$p(c_j|\mathbf{x}) = \frac{\exp\left(a_j(\mathbf{x})\right)}{\sum_{k=1}^{J}\exp\left(a_k(\mathbf{x})\right)} \tag{3.5}$$

where c_j are the J different class labels, and a_j are the MLP outputs described in the following expression:

$$a(\mathbf{x}) = \mathbf{W}^O \cdot \mathbf{\Theta}\left(\mathbf{W}^H \cdot \mathbf{x} + \mathbf{B}^H\right) + \mathbf{B}^O \tag{3.6}$$

where $\mathbf{W}$ and $\mathbf{B}$ are the MLP weights of the hidden layer (H) and output layer (O), and $\mathbf{\Theta}(z)$ is the sigmoid function.

The MLP architecture is set empirically with one hidden layer of 10 neurons, which provides good enough results to discriminate any non-linear shape without adding too much model complexity. The other MLP hyper-parameters depend on the implementation used; in this work three implementations have been used: a custom MLP in Matlab, the MLP of the built-in Matlab API *Neural Pattern Recognition* in Deep Learning toolbox [97], and the Spark MLP classifier. The hyper-parameters of the custom MLP are

listed below, for the other implementations similar parameters have been used when they were available:

- MLP architecture: One hidden layer of 10 neurons
- Neuron activation: Sigmoid
- Learning rate: 0.001
- Momentum: 0.9
- Weight decay: 0.05
- Maximum epochs: 2000

The weights are updated by training a log-likelihood objective function with a regularized back-propagation:

$$\epsilon_{LL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J} c_j(\mathbf{x}_i)\log\left(p(c_j|\mathbf{x}_i)\right) + \left(1 - c_j(\mathbf{x}_i)\right)\log\left(1 - p(c_j|\mathbf{x}_i)\right) \tag{3.7}$$

Summarizing, the Riemannian tensor metric $\mathbf{G}(\mathbf{x})$ is the Fisher information matrix $\mathbf{FI}(\mathbf{x})$ applied to a discriminative model with respect to the input space $\mathbf{x}$.

$$\mathbf{G}(\mathbf{x}) \rightarrow \mathbf{FI}(\mathbf{x}) = \sum_{j=1}^{J} \nabla_{\mathbf{x}}\log p(c_j|\mathbf{x}) \cdot \nabla_{\mathbf{x}}\log p(c_j|\mathbf{x})^T \log p(c_j|\mathbf{x}) \tag{3.8}$$

For obtaining $\mathbf{FI}(\mathbf{x})$ as a function of the MLP output estimators, the soft-max logarithms and their derivatives are needed:

$$p_j = \frac{\exp(a_j)}{\sum_{k=1}^{J}\exp(a_k)} \tag{3.9}$$

$$\log(p_j) = a_j - \log\left(\sum_{k=1}^{J}\exp(a_k)\right) \tag{3.10}$$

$$\nabla\log(p_j) = \nabla a_j - \sum_{k=1}^{J} p_k \nabla a_k \tag{3.11}$$

where $p_j = p(c_j|\mathbf{x})$, $\nabla = \nabla_{\mathbf{x}} = \frac{d}{d\mathbf{x}}$ and $a_j = a_j(\mathbf{x})$ for notation abbreviation. Now, combining eq. 3.8 and eq. 3.11 and expanding the product you obtain:

$$\begin{aligned}\mathbf{FI}(\mathbf{x}) &= \sum_{j=1}^{J}\left(\nabla a_j - \sum_{k=1}^{J} p_k \nabla a_k\right)\left(\nabla a_j - \sum_{l=1}^{J} p_l \nabla a_l\right)^T p_j \\ &= \sum_{j=1}^{J}\Bigg((\nabla a_j)(\nabla a_j)^T - \sum_{l=1}^{J}(\nabla a_j)(\nabla a_l)^T p_l \\ &- \sum_{k=1}^{J}(\nabla a_k)(\nabla a_j)^T p_k + \sum_{k=1}^{J}\sum_{l=1}^{J}(\nabla a_k)(\nabla a_l)^T p_k p_l\Bigg) p_j\end{aligned} \tag{3.12}$$

Rearranging terms, considering any variable t can be expressed as $\sum_i^J t p_i = t\sum_i^J p_i = t$, you have:

$$\begin{aligned}\mathbf{FI}(\mathbf{x}) = \sum_{j=1}^{J}\Bigg(&\sum_{k=1}^{J}\sum_{l=1}^{J}(\nabla a_j)(\nabla a_j)^T p_k p_l \\ - \sum_{k=1}^{J}\sum_{l=1}^{J}(\nabla a_j)(\nabla a_l)^T p_k p_l &- \sum_{k=1}^{J}\sum_{l=1}^{J}(\nabla a_k)(\nabla a_j)^T p_k p_l \\ &+ \sum_{k=1}^{J}\sum_{l=1}^{J}(\nabla a_k)(\nabla a_l)^T p_k p_l\Bigg) p_j \\ = \sum_{j=1}^{J}\Bigg(\sum_{k=1}^{J}\sum_{l=1}^{J}\Bigg(&(\nabla a_j)(\nabla a_j)^T - (\nabla a_j)(\nabla a_l)^T \\ &-(\nabla a_k)(\nabla a_j)^T + (\nabla a_k)(\nabla a_l)^T\Bigg) p_k p_l\Bigg) p_j\end{aligned} \tag{3.13}$$

Merging the summations you obtain the final expression of the $\mathbf{FI}(\mathbf{x})$ for the MLP:

$$\mathbf{FI}(\mathbf{x}) = \sum_{j=1}^{J}\sum_{k=1}^{J}\sum_{l=1}^{J}\nabla(a_j - a_k)\,\nabla(a_j - a_l)^T p_j p_k p_l \tag{3.14}$$

With this expression 3.14 the metric is estimated locally, computing differential distances as:

$$d(\mathbf{x}, \mathbf{x} + d\mathbf{x})^2 = d\mathbf{x}^T \cdot \mathbf{FI}(\mathbf{x}) \cdot d\mathbf{x} \tag{3.15}$$

3.2.2 Fisher pairwise distances

If there are two points *close enough*, the distance between them can be approximated as:

$$d(\mathbf{x}_A, \mathbf{x}_B)^2 \approx (\mathbf{x}_B - \mathbf{x}_A)^T \cdot \mathbf{FI}\left(\frac{\mathbf{x}_B + \mathbf{x}_A}{2}\right) \cdot (\mathbf{x}_B - \mathbf{x}_A) \tag{3.16}$$

In general, however, the points are not close. The theoretical solution for this case is to use the path integral:

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| \int_{t_A}^{t_B} \sqrt{\dot{\mathbf{x}}(t)^T \cdot \mathbf{FI}\left(\mathbf{x}(t)\right) \cdot \dot{\mathbf{x}}(t)} dt \right| \tag{3.17}$$

However the MLP density estimators, $a(\mathbf{x})$, are non-linear functions of $\mathbf{x}$ making the path integral impossible to solve analytically.

As discussed in the previous section, the distances considering the *straight line approach* can be estimated numerically, which approximates the path into a straight line that connects both points where the $\mathbf{FI}(\mathbf{x})$ is evaluated taking T samples across the path. The total distance is approximated as the sum of T small segments computed like in eq. 3.16.

$$d_T\left(\mathbf{x}_A, \mathbf{x}_B\right)^2 = \sum_{t=1}^{T} d\left(\mathbf{x}_A + \frac{t-1}{T}\left(\mathbf{x}_B - \mathbf{x}_A\right),\ \mathbf{x}_A + \frac{t}{T}\left(\mathbf{x}_B - \mathbf{x}_A\right)\right)^2 \tag{3.18}$$

The quantity of segments T can be set empirically, one option with a good trade-off in runtime versus accuracy is fixing the segments number in $T = 10$, where the segments will have a variable length depending on the Euclidean distances between $\mathbf{x}_A$ and $\mathbf{x}_B$. The other option, which is slower but the sampling would be more homogeneous for the whole data, considers a variable quantity of segments that has a fixed length defined by a global parameter, for example a fraction of the total average Euclidean pairwise distances. This process is the bottleneck of the pipeline because of the $1/2 \cdot N(N-1)$ pairwise distances to compute, at least in the single machine (CPU Intel I7, 8GB RAM) implementation with Matlab.

Next step is the *shortest path approach* which looks for an improvement of the global distances (shortening distances) applying the Floyd-Warshall algorithm across the Fisher manifold. This kind of algorithm is based on weighted graphs. In our case, the nodes are the points and the edges are the pairwise distances, creating a fully connected graph. For the implementation, *MatlabBGL* has been used from the library of David F. Gleich.

With the *shortest path approach* the pairwise distances are assured to be the shortest possible, at least within the manifold obtained with the *straight line approach* pairwise distances. The higher the density points when computing pairwise distances, the higher the information of the manifold, producing more accurate global distances.

3.2.3 Community finding in a similarity network

This section describes the implementation in Matlab of the community finding algorithm based on modularity spectral optimization [12]. Before this, an important step is the transformation of the manifold distances into a similarity network using a Gaussian kernel of eq. 2.18. The network is defined by the adjacency matrix A_{ij} which is symmetrical with diagonal elements forced to be zero. Elements with small similarities are rounded to zero, by default $A_{ij} < 10^{-5}$.

The locality parameter σ_G is a free parameter that needs to be adjusted. Two methods are proposed: the first one measures the faithfulness of the network predictions compared to the MLP. This process implies running the community finding algorithm for a range of σ_G and evaluating the network for each value. The second is a heuristic approach based on the average intra-label distances in the Fisher manifold. The first method will be used for the Matlab implementation, and the second method for the Spark implementation (section 3.3) as this approach requires less computational cost.

3.2.3.1 Length scale based on the faithfulness of the network predictions

Three criteria have been used to measure the faithfulness of the network predictions. The method generates different networks using a range of length scales comprised into the maximum and minimum manifold distance, and then each network is visually assessed to select the appropriate σ_G using a plot with the following criteria: KL divergence, Cramer's V statistic and McNemar's test.

KL divergence:
A new classifier model can be estimated based on the network combining similarities with the MLP outputs, the scores $a_k(\mathbf{x})$. Applying a weighted average over the scores, where the weights are the similarities, a new score $a'_k(\mathbf{x})$ is obtained estimated by the network:

$$a'_k(\mathbf{x}) = \frac{1}{\sum_j^N A_{ij}} \sum_j^N A_{ij} a'_k(\mathbf{x}_j) \tag{3.19}$$

then it can be used as an argument in the soft-max function 3.5 to obtain the probabilities $p'(c_k|\mathbf{x})$.

The network model can be compared with the original MLP model using the KL divergence, showing how well the network predictions replicate the MLP predictions:

$$\varphi_{KL} = -\frac{1}{N}\sum_{i}^{N}\sum_{k}^{J} p(c_k|\mathbf{x}) \log\left(\frac{p'(c_k|\mathbf{x})}{p(c_k|\mathbf{x})}\right) \tag{3.20}$$

If $\varphi_{KL} = 0$ this means the network perfectly reproduces the estimates of the MLP. This is a good method to clearly indicate an appropriate value for the length scale. When σ_G is small, $\varphi_{KL} \approx 0$ because the network only considers the closest neighbours for the predictions. However, as σ_G increases, at some point φ_{KL} will also start to increase, at the point where σ_G starts to increase considerably is the preferred value for σ_G. In other words, the objective is to select the largest value of σ_G whilst keeping $\varphi_{KL} \approx 0$.

Cramer's V statistic:
This statistic [71] evaluates how the communities match the true labels, with $\varphi_{CV} = 1$ representing complete concordance and $\varphi_{CV} = 0$ no association at all. It is based on the contingency table formed by the communities and the class labels. Here the objective is opposite to the φ_{KL} case, here the objective is to select the largest σ_G that also has $\varphi_{CV} \approx 1$.

$$\varphi_{CV} = \sqrt{\frac{\chi^2}{N \cdot \min\left(Com - 1, J - 1\right)}} \tag{3.21}$$

where χ^2 is the chi-squared statistic obtained from the table, and Com and J are the number of communities and class labels respectively.

McNemar's test:
McNemar's test [98] is based on the difference between the prediction made with the probability estimated by the FI network and the original probability estimated by the MLP. The null hypothesis is that both models are not significantly different. The objective is to select a σ_G in a range where φ_{MN} is high enough to be greater than the *p-value* of the statistical test, to assure that both models are not significantly different.

$$\varphi_{MN} = \frac{|n_A - n_B| - 1}{\sqrt{n_A + n_B}} \tag{3.22}$$

where n_A are the number of errors made by the network and not by the MLP, and n_B are the number of errors made by the MLP but not by the network.

3.2.3.2 Length scale based on the heuristic intra-labels manifold distances

This method is based on the average pairwise distances of the manifold that belong to the same label, let us call these the intra-label average distances:

$$\mathrm{dist}_{intraLab}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\sum_{i=1}^{N} \sum_{j\ni(j>i)\wedge(L_i=L_j)} \mathrm{dist}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\sum_{i=1}^{N} \sum_{j\ni(j>i)\wedge(L_i=L_j)} 1} \tag{3.23}$$

This intra-label distance represents the average distance within the manifold regions that belong to the same class label. Using the Gaussian kernel to measure similarities, $A_{i,j} \in [0,1]$, the constraint that this intra-label distance (or a fraction of that distance) have to be equal to a reasonably high value of the similarity measure can be imposed: $A_{th} > 0.5$. Let say $A_{(i,j)\in \mathrm{intraLab}} \rightarrow A_{th} = 0.75$ for instance.

$$A_{(i,j)\in \mathrm{intraLab}} = \exp\left(-\frac{r \cdot \mathrm{dist}_{\mathrm{intraLab}}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\sigma_G}\right)^2 \leftarrow A_{th} \tag{3.24}$$

Imposing these conditions, σ_G becomes completely determined within a small range of values. A multiplying factor $r \in [0,1]$ has been added to the distances, to modulate the granularity of the network. Empirical good values are $r \in [0.2, 0.5]$. With $r = 0.5$ the community finding algorithm obtains approximately as many communities as there are different class labels. With $r = 0.2$ between one and two times the number of different class labels are approximately obtained.

$$\sigma_G \leftarrow \frac{r \cdot \mathrm{dist}_{\mathrm{intraLab}}\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\sqrt{\ln\left(A_{th}\right)}} \tag{3.25}$$

Once the σ_G is set, the adjacency matrix $A_{i,j}$ is calculated, then follows the community finding algorithm to obtain the community membership list. Additionally the prototype of each community can be computed, where the prototype is the node of highest influence within the sub-graph generated by the community. The prototype can be estimated using the concept of centrality in graph-theory. There are several centrality indicators, for this work the closeness centrality or the eigenvector centrality was used.

The next step is to visualize the FI network with the communities and prototypes that have been obtained.

3.2.4 Low-dimensional representation

This section introduces embedding of the Fisher manifold into a low-dimensional space for visualizing the structure of the Fisher manifold, ideally in a two or three dimensional space. The basic idea is to transform the pairwise distances of the Riemannian manifold into coordinates, embedded in a Euclidean space. Most methods require approximations to achieve a low-dimensional representation of the Riemannian manifold. These approximations are acceptable so long the margin of error is controlled. Two methods have been implemented, each one with its own specific advantages.

3.2.4.1 Sammon mapping

For Sammon mapping it would be a two-dimensional representation. It tends to be more accurate at preserving the local distances, but the global distances are distorted. This method is very useful for representing the communities as prototypes. However, given Sammon mapping is a non-linear method that forces projections into a two-dimensional space, sometimes the Riemannian manifolds cannot be embedded only in this space, producing significant distortion, overall in the long distances, without knowing the level of error in the global distances. For this reason, the Sammon mapping is good for visualizing the communities and the relative distances of the Fisher manifold, but is not a good method for generating a faithful Euclidean embedding where projective clustering methods can be applied.

In figures 3.2 and 3.3 an example of Sammon mapping is presented using the pairwise distances of the well-known Iris dataset. Figure 3.2 embeds a network based on Euclidean distances, whilst figure 3.3 embeds a Fisher Information network. The FIN is based on the MLP classifier with 97% accuracy, hence the Fisher manifold can separate the three classes very effectively (Setosa, Versicolour and Virginica). In both cases, the communities have been found using a heuristic-based length scale. The communities of the Euclidean distances are mixing the real classes, however in the Fisher manifold, the classes are well separated, except for a few observations. With Sammon mapping the communities can easily be visualized in a two-dimensional representation with a similar scale for both axis, although the global distances are distorted.

3.2.4.2 Classical Multidimensional Scaling

The cMDS has a complementary advantage to Sammon Mapping. It is a good method for preserving the global structure of the manifold, however, it needs more dimensions to properly embed a Riemannian manifold with the advantage of gaining mapping accuracy.

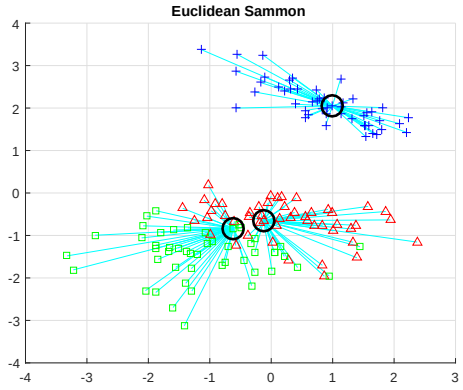


FIG. 3.2 – Iris Euclidean with Sammon mapping

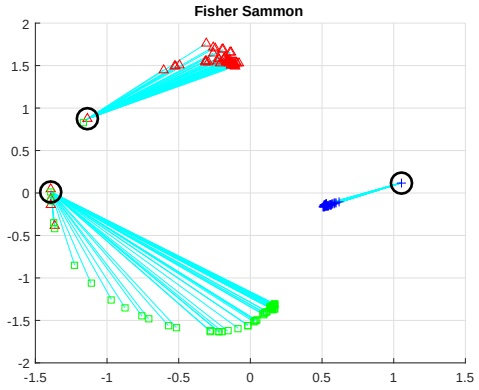


FIG. 3.3 – Iris FIN with Sammon mapping

The interesting point is that the method gives access to the eigenvalues associated with each eigenvector of the Euclidean embedding, measuring the relative importance of each dimension, and therefore discarding the less relevant dimensions beyond some threshold of the cumulative sum of the eigenvalues. These eigenvalues represent the information (variance) carried in each eigenvector (dimension), they are sorted in descending order which provides a clear indication of how many dimensions are needed to effectively map the Fisher manifold.

The cMDS represents a more faithful structure of the Fisher manifold than the Sammon mapping, generating a Euclidean space that can be used for projective clustering methods. Figures 3.4 and 3.5 show the Iris example with the cMDS embedding, at first it looks similar to the Sammon mapping, but inspecting the axis more closely they do not have the same scale, the eigenvector decomposition allowing the relative importance of each dimension to be observed.

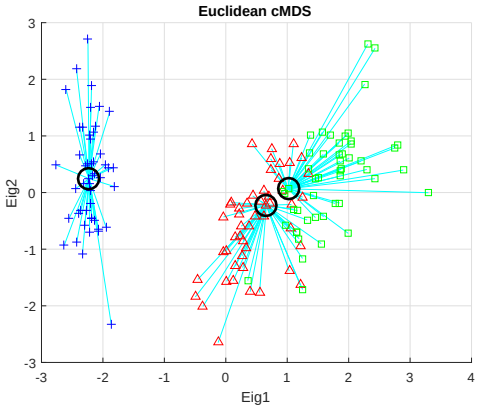


FIG. 3.4 – Iris Euc. with cMDS

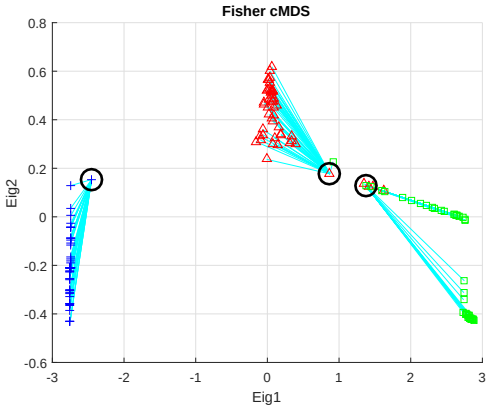


FIG. 3.5 – Iris FIN with cMDS

In figures 3.6 and 3.7, the same cMDS embedding is depicted with the first three eigenvectors only, and fixing the axis scales for the three dimensions. In figure 3.7 it can be

seen how the Fisher manifold lies in practically one dimension.

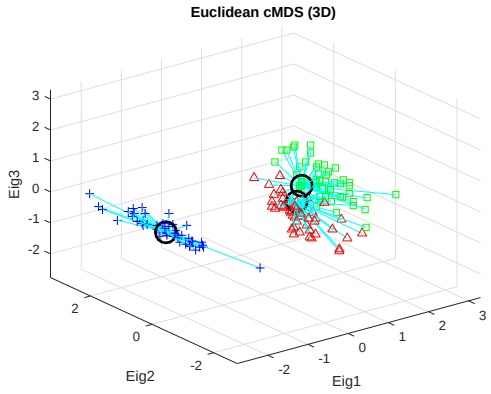


FIG. 3.6 – Iris Euc. with cMDS in 3D fixed axis

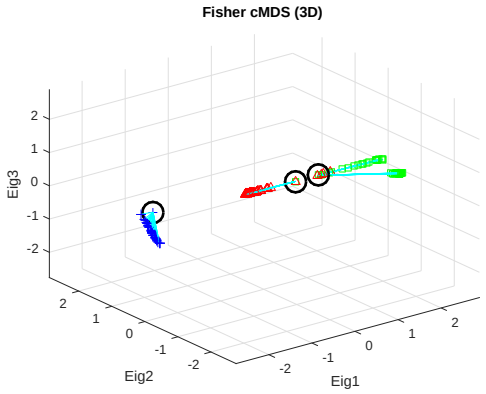


FIG. 3.7 – Iris FIN with cMDS in 3D fixed axis

Plotting the eigenvalues and their cumulative sum in figure 3.8, one can appreciate that, for the Iris dataset, the first eigenvalue contains almost all the variance of the Fisher manifold. In these cases where the first eigenvector is clearly dominant, a histogram could be used to represent the first eigenvector projection, allowing discrimination of the labels by its distribution. Figure 3.9 shows the joint probability based on the density histogram multiplied by the class prevalence: $P(X, C) = P(X|C) \cdot P(C)$

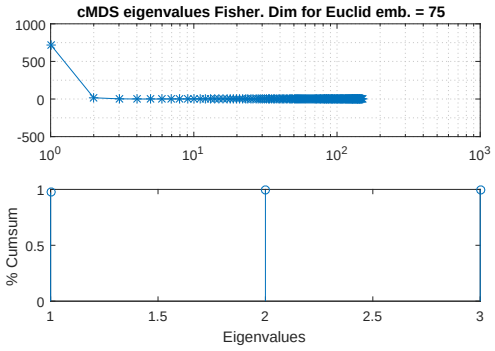


FIG. 3.8 – Eigenvalues of cMDS Iris FIN

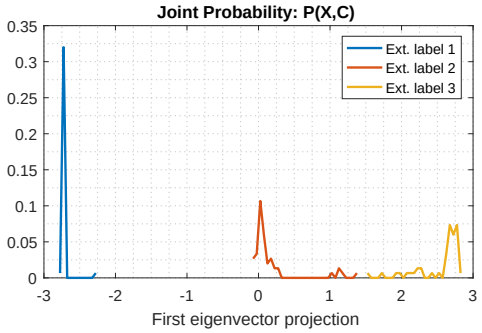


FIG. 3.9 – Iris FIN histogram of first eigenvector projection

3.2.5 Community profiles

One additional step is to analyse the community profiles in order to highlight if the communities are characterized by any particular features in the input space. This step basically builds a table of k communities and n features where each feature is standardized with respect to the average feature considering the whole data:

$$\text{profile}_{com_k}(x_n) = \frac{\mu_{com_k}(x_n) - \mu(x_n)}{\sigma(x_n)} \tag{3.26}$$

x_n being the nth feature of the input space, μ_{com_k} the mean value of the observations that belong to community k, μ the mean feature value and σ its standard deviation.

These community profiles are obtained with the information extracted from the Fisher manifold and can be compared with the profiles based on the class labels (computed analogously). Usually there are significant differences between community and class label profiles, with the community profiles usually having some features with values considerably different from the mean. However the class label profiles do not necessarily have to have a characteristic profile.

The community profiles will be used in Chapter 6 to describe the customer shopping basket profiles in section 6.2.10, and the music profiles in section 6.3.4.4.

3.3 Scalable implementation of Fisher Information Networks

The objective of this section is to reduce the pipeline runtime for larger datasets. The main problem of FIN is that the algorithm is not scalable because at least one part of the procedure heavily depends on pairwise distances, as explained in previous section 1.3.1, and each Riemannian pairwise distance needs relatively expensive computations at two levels, one local and another global. Locally, the Fisher metric needs to be evaluated several times across a straight-line path. Subsequently, the global pairwise distances are improved by computing the APSP across the manifold taking into account all the pairwise distances computed locally.

Unfortunately, although big data tools are employed, the FIN pipeline does not work for typical big data environments, such as $N \approx 10^5 - 10^6$, because of the quadratic dependency $O\left(N^2\right)$ of the algorithm. The runtime will eventually grow to a point beyond which a big data environment is not feasible. Therefore, the objective is to use big data tools to be able to tackle greater sample sizes than the CPU-*single machine* version of the algorithm, the sample limit will depend on the computer performance, but is roughly around $N \approx 10^4$. For instance, the procedure was tested in a Hadoop cluster of 150 nodes with 128GB RAM and 48 cores each, using 15% of the cluster capacity, and for sample sizes of $N \leq 2 \cdot 10^4$, the runtime was less than eight hours.

This section will introduce some common big data tools, describe the stages of processing implementation in Spark and will show some interesting data examples of the FIN with larger sample sizes.

3.3.1 Introduction to Big Data Framework

The big data paradigm has had an enormous impact over the last 10 years. The fast growth of data generation, together with the huge advances in computing power and technology, have been the main catalyst for the launch and development of the Big Data era. Big data has been characterised by a number of "Vs", including Volume, Veracity, Velocity and Variety, and to the more recent additions of: Viability, Visualization and Value.

The purpose of this section is to present the Big Data tools that will speed up the pipeline runtime, under certain external constraints imposed by the collaborating retail company. Investigating the procedure bottlenecks, it is apparent that the main challenge are the pairwise distances, and the best way to tackle this problem is parallelization. What also needs to be taken into account is the data volume and whether it can be held in memory, which is mainly due to the APSP pipeline step.

Parallel computing tools fall into two main groups: single-machine parallel computing and distributed-computing:

Single-machine parallel computing is focused on multi-thread CPU cores and the General-Purpose computing on Graphics Processing Units (GPGPU). The GPGPU offers a level of parallelization much higher than multi-thread CPU cores, although the CPU chips are faster, the GPGPU are built with a larger number of graphics chips. The GPGPU paradigm is based on handling the data as if they were images, where the parallel processing is performed between one or several GPUs and a CPU. Nowadays there are many high level programming languages and software packages or application programming interfaces (APIs) that can relatively easily implement parallel computing, for instance the Nvidia CUDA [99] platform or the TensorFlow [100] open-source software for GPGPU, also Matlab has the Parallel Computing Toolbox [101]. This single-machine category is usually associated with High Performance Computers (HPC), but it does not necessarily need to be a single computer, it can be a server or a HPC cluster, or any other configuration that differs from multiple machines with a distributed location (networked computers) which is distributed computing.

Distributed computing is generally characterized by: lack of a global clock, concurrency of components, and independent failure of components. The power of this configuration relies on the large number of independent computers that can run many parallel processes (tasks) providing for failure tolerance and redundancy, like the executors in the Java Virtual Machines (JVM) created in the Hadoop ecosystem. One of the most relevant paradigms of big data analysis is the MapReduce paradigm [102] developed by Google, later becoming open-source and under the Apache Foundation, integrated into

the Hadoop project [5]. Apache Hadoop is a family of open-source software that forms an ecosystem. One of the main elements is a distributed storage system known as Hadoop Distributed File System (HDFS), which was inspired by Google File System [103]. Also, Hadoop includes the Hadoop MapReduce implementation for large-scale data processing, Hadoop YARN which is a resource manager and applications scheduler, and Hadoop Common with the libraries and utilities. The Hadoop ecosystem is designed to operate in distributed computer clusters built from commodity hardware, where the failure tolerance relies on data redundancy and the cluster self-governance controlled by YARN managing the nodes as workers or masters, which is very well suited to cloud-computing.

Originally, in the Hadoop ecosystem, most of the machine learning (ML) implementations were based on MapReduce, for example the Apache Mahout project [104]. However the main disadvantage of MapReduce is the disk-oriented design which slows the machine learning algorithms with high iteration dependency and the high load on disk writing/reading operations. This is the main reason why MapReduce has been superseded by the Apache Spark project [6], where Spark bases the workload on the nodes RAM memory allowing for fast iterations for ML algorithms. Apache Spark was originally developed at the University of California, Berkeley's AMPLab, in 2009, being donated to the Apache Software Foundation in 2013. Nowadays Spark has become the staple for big data analytics offering a unified engine with multiple modules that cover most ML implementations: Spark SQL, Spark Streaming, MLlib and GraphX. Spark can run as a top layer within the Hadoop ecosystem. It manages the data using the resilient distributed dataset (RDD), a read-only (immutable), partitioned collection of elements that can be operated on in parallel. Newer Spark versions have improved the RDD into Dataset and DataFrames for SQL, where one of the main improvements relies on the DAG scheduler and the query optimizer under the lazy-evaluation of Spark. The RDD operations can be divided into actions and transformations, Spark only performs computations on RDD actions, hence it collects sequential RDD transformations and then are executed with the optimized order (DAG scheduler) when the next RDD action triggers the computation. The order in which the operations are evaluated has a strong impact in the runtime performance, overall in operations that involve data shuffling across different cluster nodes.

From the parallel computing tools discussed, Hadoop + Spark ecosystem has been chosen for a number of reasons. Even when it is not the fastest option when compared with a high performance computer based on GPGPU, it does offer scalability and versatility in cloud computing, allowing the cluster size to adapt as required. In addition, it is failure-tolerant and provides a robust file management system that combines with no-SQL databases based on Hadoop, like Apache HBase, Cassandra or MongoDB. This is also the set up the collaborating retail company prefers.

3.3.2 Methodology of Spark implementation

The Spark implementation of the FIN procedure uses the ML libraries every time it was possible, however, a number of custom functions have also been written. DataFrame-based implementations were the preferred option. A Data-Frame is a Dataset organized by columns, where a Dataset is a distributed collection of data, with the benefits of the RDDs and employing Spark SQL's optimized execution engine.

This is the list of steps in the Spark implementation:

1. Data preprocessing with Spark ML library based on DataFrames.
2. MLP from DataFrame-based Spark ML library to obtain the neuron weights.
3. Fisher metric and pairwise distances with the straight-line approach using User Defined Functions (UDF) based on DataFrames.
4. Shortest path for distances optimization using Dijkstra algorithm approximations
5. Community finding using the Power Iteration Clustering (PIC) built in the RDD-based Spark ML library.

In Spark, the number of partitions needs to be set, in which the RDD are parallelized taking into account the number of executors (cluster size) and the sample size. Too many partitions produce excessive overheads and too few does not take full advantage of the parallelization. The trade-off is determined empirically by increasing the number of partitions until the first overhead indicators appear. However, a good approximation of the partitions number is two or four times the total number of cores in the cluster.

In the FIN pipeline, two different partition levels p_1 and p_2 are set, the first partition level p_1 is derived from the input sample size, where the initial RDD is partitioned according to the number of observations, most of the stages of processing operations work with these partition levels, except for the pairwise distances. The pairwise distances are treated like a list of $0.5N(N-1)$ elements, and for these computations a new RDD is created with the higher partition level p_2. There is no a rule of thumb to determine $p2$, but a good range is $p_2 \in \left[4p_1, 0.5p_1^2\right]$, setting a linear dependence for small sample sizes and a quadratic dependence for relatively large sample sizes. It is also advisable to set a maximum p_2 value (threshold) that depends on the cluster size in case of large $p1$ values.

3.3.2.1 Computing the Fisher Information metric

For computing the Fisher Information metric, the MLP neurons weights are needed. The weights carry all the information of the Fisher metric and the manifold shape is completely determined by them. Additionally, if the input data is standardized, the average weights values are independent of the input dataset.

Using the MLP of the Spark ML library has some disadvantages, for example, the algorithm cannot be customized to include other options. One example of this is that it does not implement the use of a regularization term based on weight decay, which is crucial to optimize the neuron weights with small values. This is probably because the MLP is designed to deal with big data, where the risk of over-fitting is reduced. There are, instead, two options for optimizing the neuron weights: mini-batch gradient descent or L-BFGS [105], with the latter being the one used by default.

L-BFGS stands for Limited-memory of the Broyden–Fletcher–Goldfarb–Shanno algorithm, and is based on the family of quasi-Newton methods. This option makes the MLP training converge faster and the performance is slightly better than gradient descent. However this method tends to assign non-standardised dissimilar values to the neuron weights, some of them being too large, $O(10^2)$. Large neuron weights produce an unstable Fisher metric in regions with high probability gradients, making the metric matrix almost a degenerate matrix and producing some distances that become practically zero and others that are too large, creating a twisted Fisher manifold. Therefore, to obtain a homogeneous Fisher manifold, with well-balanced distances, it is important the values of the neuron weights are similar with values close to zero. This is the main reason to reject L-BFGS optimization and use the mini-batch gradient descent, despite its generally worse performance. With the gradient descent option, in order to achieve convergence, more iterations are needed, and also an increased learning rate for highly non-linear data, but the neuron weights are within an acceptable range.

Another option for obtaining MLP neuron weights when the input sample is not large $O(10^5)$, is to use the customizable Matlab MLP and then export the weights to Spark to carry on with the Spark pipeline. The clear disadvantage of this is having to deal with separate environments, i.e. Matlab and Spark.

Once the neuron weights are calculated, a Spark DataFrame is created with the list of all pairwise distances, using the higher partition level p_2. Then, custom functions are defined to compute the Fisher pairwise distance using the straight-line approach of section 3.18, where the input function is a pair of points and the output function is the Fisher distance. Within the function, the Fisher metric is evaluated in regular intervals between the points and the Fisher distance is estimated as a sum of small segments. This

custom function is transformed into a UDF to be applied to a DataFrame pairwise list, parallelizing the distance computation into p_2 partitions. Bear in mind the partitions are not evaluated simultaneously, they are evaluated by the executors of each JVM, where each executor has assigned a certain number of cores that are able to deal with simultaneous processes. The executors' configuration is set in the cluster creation, but the partitions are set within the Spark session.

As previously mentioned, the main procedure bottleneck are the pairwise distance computations, and the UDF applied to the DataFrame is the tool employed in Spark to speed up these computations. The runtime depends on the cluster size and its performance, however the parallelization increases and the runtime decreases linearly with the cluster size and cores, but the pairwise distance list increases quadratically with the sample size. Therefore, eventually, there will be a sample size with an excessive runtime for our application.

3.3.2.2 Shortest paths approximations

This section also presents a bottleneck in the pipeline, although it is not as critical as the bottleneck described in the previous section. Here the main purpose is to optimize the pairwise distances computed for the straight-line approach using the Fisher manifold to find a shortest path that reduces the distance.

In Matlab, the Floyd-Warshall algorithm was used to compute the shortest path from all pairs at once (APSP). The Matlab implementation is designed to deal with data held in memory, therefore under this requirement, the algorithm is considered to be fast and efficient.

However, in Spark, everything is designed for distributed memory, and a stable implementation of the Floyd-Warshall algorithm has not yet been implemented, although there are some related works such as [106]. One of the limitations of the APSP algorithms in distributed computing is that the graph is already distributed, parallelizing the message passing of one source node among the other graph nodes in a fully connected graph. Therefore, it is difficult to implement another level of parallelization, forcing the message passing of all sources to be implemented sequentially.

To overcome this problem the Dijkstra algorithm can be used instead, which is a single-source shortest path algorithm (SSSP) and has been implemented in Spark within the GraphX API. However, approximations to estimate all sources have to be made using a single-source algorithm. Although the direct solution could be to sequentially apply the Dijkstra algorithm to all sources, when there are many sources (observations) this approach becomes intractable. Two approximations are proposed:

- Approx. 1: Prototypes shortest paths.
- Approx. 2: Shortest paths pivoting with random points.

Approx. 1: Prototypes shortest paths

This approach applies the PIC method over the Fisher manifold for obtaining clusters and their prototypes, then instead of computing the shortest path over all pairwise distances, the shortest path is only computed between prototypes using only the prototype network. Therefore, the number of nodes (sources) to apply the shortest path is drastically reduced, the Dijkstra algorithm can then successfully applied sequentially over all prototypes. Constructing the Fisher pairwise distances as a graph G of N nodes with N^2-N edges defined by the distances of the straight-line approach, the graph is reduced to K nodes of K^2-K edges, where K is the number of prototypes, being $K \ll N$.

The intra-cluster distances, i.e. within the same cluster, are estimated as the distances computed in the previous section with the straight-line approach, dist_{SL}:

$$\text{dist}_{intra-cluster}(i,j) = \text{dist}_{SL}(i,j) \tag{3.27}$$

The inter-cluster distances are estimated as the sum of three distances: SL distance from node i to its prototype k_i, geodesic distance between prototypes k_i and k_j, and SL distance from node j to its prototype k_j:

$$\text{dist}_{inter-cluster}(i,j) = \text{dist}_{SL}(i,k_i) + \text{dist}_{geod}(k_i,k_j) + \text{dist}_{SL}(k_j,j) \tag{3.28}$$

This approximation is analogous to an underground rail network, where the geodesic prototype distances represent the railway stations, and the node to prototype SL distances acts as people walking to the stations. Only the station (prototypes) distances are optimized with the shortest path algorithm.

There are some risks related to the loss of information in the Fisher manifold if the number of prototypes are reduced too much when computing the Dijkstra algorithm, because the optimal path is found only through the prototype distances.

The approximation is not as efficient as the optimal distances with the Floyd-Warshall algorithm, but it roughly captures the main structure of the manifold. The only considerable problem is related to the Spark data architecture. This procedure contains several DataFrame JOIN operations by indices that probably belong to different partitions, and this kind of transformations with wide dependencies can be slow in Spark when data is

shuffling among cluster nodes. Therefore, a second approximation method was devised (see section 3.3.2.2).

Approx. 2: Shortest paths pivoting with random points

This approach tries to reduce the runtime by avoiding finding clusters and prototypes. The basic assumption is that close points in the Fisher manifold are well estimated by the SL distances, and distant points are more likely to require the use of the shortest path algorithm across the whole manifold to compute the geodesic distance.

The Dijkstra algorithm obtains the shortest paths of a single source with respect to the rest of the nodes, therefore selecting k' random nodes for applying the Dijkstra algorithm sequentially, the information can be used to estimate geodesic distances from any nodes pivoting over all of this k' nodes and then select the minimum of these k' distances:

$$\text{dist}_{pivot}(i,j) = \min_{k'}\left(\left[\text{dist}_{geod}\left(i,k'\right) + \text{dist}_{geod}\left(k',j\right)\right] \forall\, k'\right) \tag{3.29}$$

Thus assuring the minimum distance between any two points when comparing dist_{pivot} with dist_{SL} the minimum distance is selected, if they are close points it is probable that $\text{dist}_{pivot} > \text{dist}_{SL}$.

$$\text{dist}(i,j) = \min\left(\text{dist}_{pivot}(i,j), \text{dist}_{SL}(i,j)\right) \tag{3.30}$$

With approx.2 better results are obtained compared to approx.1 with reference to the optimal APSP solution with the Floyd-Warshall algorithm. In addition, the runtime is considerable reduced because some of the DataFrame JOIN operations have been replaced for MapReduce operations, reducing the data shuffling across the cluster nodes.

Although this method is scalable in Spark, for those cases where the data fits into memory able to run on a single-machine, the runtime of the Spark version is much greater than running the Floyd-Warshall algorithm in Matlab.

3.3.2.3 Communities with Power Iteration Clustering

In the previous Subsection 3.3.2.2, the PIC [107] was introduced which is a spectral clustering method implemented in the RDD-based Spark ML library and able to deal directly with the adjacency matrix based on similarities as the input data. This is of benefit as the Fisher manifold is described as distances instead of Euclidean coordinates, and PIC can deal with Riemannian distances when they are transformed similarity

measures. Here a Gaussian kernel with the length scale determined heuristically is used with the method described in section 3.2.3.2.

One difficulty in using the algorithm is the requirement to provide the number of clusters as an input parameter, meaning there is no automatic detection for number of the communities. The consequence is if the number of clusters stated is too large, the algorithm will find clusters with very low membership.

Once the communities are obtained, the community profiles are computed following the same procedure in section 3.2.5, but using Spark DataFrames with *groupBy* functions to compute the profiles.

3.3.3 Runtime problems and the hybrid solution

The Spark implementation has been tested in different cluster configurations:

1. High performance computer with 768GB RAM and 32 cores in Spark standalone mode.
2. Cloud computing with Amazon Web Services using different configurations up to using: 20 nodes of 32GB RAM and 8 cores each, or 5 nodes of 244GB RAM and 32 cores each.
3. Cluster of the retail company, using up to 15% of its resources: 150 nodes, 128 GB RAM and 48 cores each.

The third option being the most powerful cluster and with the smallest pipeline runtime.

As mentioned before, the Spark work-flow has two main bottlenecks, $Bn1$ and $Bn2$: the first $Bn1$, is in the pairwise distances with the straight-line approach, and the second $Bn2$, is in the APSP with the second approximation 3.3.2.2. The most important factor affecting runtime is the sample size, but it is also affected by the number of features, and number of categories in the class labels. In the Spark implementation, the two bottlenecks take approximately the same amount of time, slightly more for $Bn2$.

Taking the retail dataset as a reference described in detail in section 6.2 of Chapter 6: it has 35 features, 4 categories in the class labels, and a sample size up to 50K observations. This information can be used to estimate the maximum sample size to perform the work-flow in an overnight or less than eight hours.

Table 3.1 shows approximated runtime for different cluster configurations. In Matlab implementation, $Bn1$ refers to the SL distances, and $Bn2$ refers to the Floyd-Warshall

algorithm, which is not really a bottleneck if compared with $Bn1$. The standard PC has an I7 processor with 8GB RAM and shows the runtime for the implementations in Matlab. The HPC configuration uses a hybrid solution, where $Bn1$ is implemented in Spark taking advantage of the 32 CPU cores in standalone mode, and $Bn2$ is implemented in Matlab. The AWS and the big clusters (retail company cluster) use both the Spark implementation.

TABLE 3.1 – Approx. runtime for different cluster configurations

	5K obs		**10K obs**		**15K obs**	
Computer config.	Bn1	Bn2	Bn1	Bn2	Bn1	Bn2
Standard PC	5d	5'	-	10'	-	20'
HPC	12h	5'	2d	10'	-	20'
AWS cluster	1h	1.5h	3h	5h	12h	20h
Large cluster	30'	45'	1.5h	2.5h	5h	7h

Inspecting the table, for more than 15K observations the pipeline runtime takes too long, even when using a large cluster. This implies that the implementation is not really capable of dealing with "big data", although there are 112.5M different pairwise distances to compute. For sample sizes lower than 15K, where the data fits into memory of a standard computer, the best option is to use a hybrid solution, where the Spark implementation is the best option to speed up $Bn1$, and the Matlab implementation is better to tackle $Bn2$ which provides the exact shortest paths with the Floyd-Warshall algorithm.

With the hybrid solution, the Spark implementation is only needed for computing the SL distances, the rest of the procedure can be implemented in Matlab as described in section 3.2, including the MLP. If there is no access to a large cluster, an easy way to use Spark is through cloud computing services (AWS, Azure, Cloudera, Hortonworks, etc.) for creating a temporary cluster adjusted, in this work AWS ElasticMap Reduce clusters were used.

Therefore, the hybrid approach only considers the fastest combination of the method to improve the runtime, which is a good solution to launch research experiments. However this solution, even if it is the fastest, may not be appropriate for a production deployment where all the algorithms should be integrated in the same environment.

3.4 Case studies

The methodology to obtain the Fisher manifolds has been applied on three case studies: aneurysm, nested spirals and exotic physics particles data. The first one does not use

the scalable implementation because the dataset is small enough, on the other hand the other two cases use the scalable implementation.

In all cases, the clustering task has been made using spectral clustering, choosing the left path in the step 6 of the pipeline diagram, figure 1.1 of Chapter 1. The other path with PQC will be introduced later in next Chapter 4, and both paths will be compared in Chapter 6 with two case studies. In addition, for the cases of this chapter it is not necessary to apply PQC to obtain a good analysis of the manifold. When the manifold is one-dimensional, with density histograms it is enough to evaluate the clusters.

3.4.1 Aneurysm case study

This subsection shows the results of applying the FIN to a real-world data case based on cerebral aneurysms [108]. The data combines traditional clinical records with complex features obtained by computational analysis [109]. From each patient it is known whether the aneurysm ruptured or not, and this variable is used as the target label. The main objective is to obtain a rupture risk stratification of cerebral aneurysm and, by applying the FIN, one expects to find communities from which patient profiles can be defined, with a higher predictive power than a traditional binary classifier. The Fisher manifold has to stratify the transition from regions of high rupture risk to safer regions, where the rupture is less likely.

A cerebral aneurysm is a complex disease which incurs severe damage if a rupture ensues, leading to 60% mortality rate in such cases [110]. The two common treatments for ruptured aneurysm are surgical clipping or endovascular coiling. However, not all aneurysms cause a rupture, and with the treatment being a risky operation, it is only recommended for aneurysms with a high risk of rupture.

The objective of this analysis is to identify patient profiles (communities) with a high risk of rupture where the treatment would be advisable for patients before the rupture occurs, and also identify other low-risk profiles where alternative treatments, such as drugs, can be employed.

Using the FIN procedure, the Fisher manifold is generated by the MLP. Given that this is a binary classifier, a low-dimensional manifold is expected. This manifold contains the information needed to discriminate the rupture, and the patients can be ranked by the FI metric similarity enabling stratification by communities of different rupture risk.

3.4.1.1 Data description and feature selection

The data consists of 180 patients' clinical records combined with basic shape descriptors of the aneurysm [111, 112] and a set of complex features obtained by computational analysis [113]. The complex features describe the surface and volume of the Zernike Moment Invariants (ZMI), which are expressed in 120 principal components from PCA, with only the top 20 PC considered. The clinical records are eight categorical variables that include the well-known risk factors: age, smoking status, hypertension status, location of aneurysm, laterality of aneurysm, status of aneurysm and aneurysm type. There are eight basic shape aneurysm descriptors: volume, surface, neck surface, neck width, depth, aspect ratio, and non-sphericity index.

The ratio of the number of variables to observations (patients) is very likely to lead to overfitting, and it is also likely that many of the ZMI components are redundant. To tackle this problem a feature selection by means of a CI-map is performed, which was constructed incorporating the target variable (rupture) and all the features, previously categorized (binary). The idea being to select only those variables that have a first order (direct edge) or second order (two edges distance) connection to rupture. For more statistical robustness, the CI-maps were resampled with replacement (similar to bootstrapping) to create a frequency list of those variables that are more often directly connected with rupture.

From the top 12 variables a subset of 8 was selected, which resulted in highest MLP accuracy (64% on the test data) with the lowest amount of features as possible. These top 8 features most associated with rupture were: location of aneurysm, hypertension, non-sphericity index, rem-PC6, ZMI surface-PC6 and ZMI surface-PC4.

As a pre-processing step, the categorical variables were transformed with 1-N encoding and the continuous data was standardized (z-score).

3.4.1.2 Results and discussion

The first step is to train the MLP, in this case has been used the built-in Matlab API *Neural Pattern Recognition* [97], with one hidden layer of 10 neurons as mentioned in section 3.2.1. The MLP performance is depicted in a contingency table showing the scores in the following template 3.2:

In multinomial cases, the sensitivity, specificity, PPV and NPV are computed as *One versus All*, for instance if there are 3 class labels, where TP_i are the true positives of $class_i$, its sensitivity $Sens_i$ is:

TABLE 3.2 – Template for showing the MLP results.

	Target 1	*Target 2*	
Predicted 1	**True Positive**	**False Positive**	PPV
Predicted 2	**False Negative**	**True Negative**	NPV
	Sensitivity	Specificity	Accuracy

$$Sens_i = \frac{TP_2}{TP_1 + TP_2 + TP_3} \tag{3.31}$$

The data is divided in 70% training, 15% validation and 15% test. The results are shown in figure 3.10

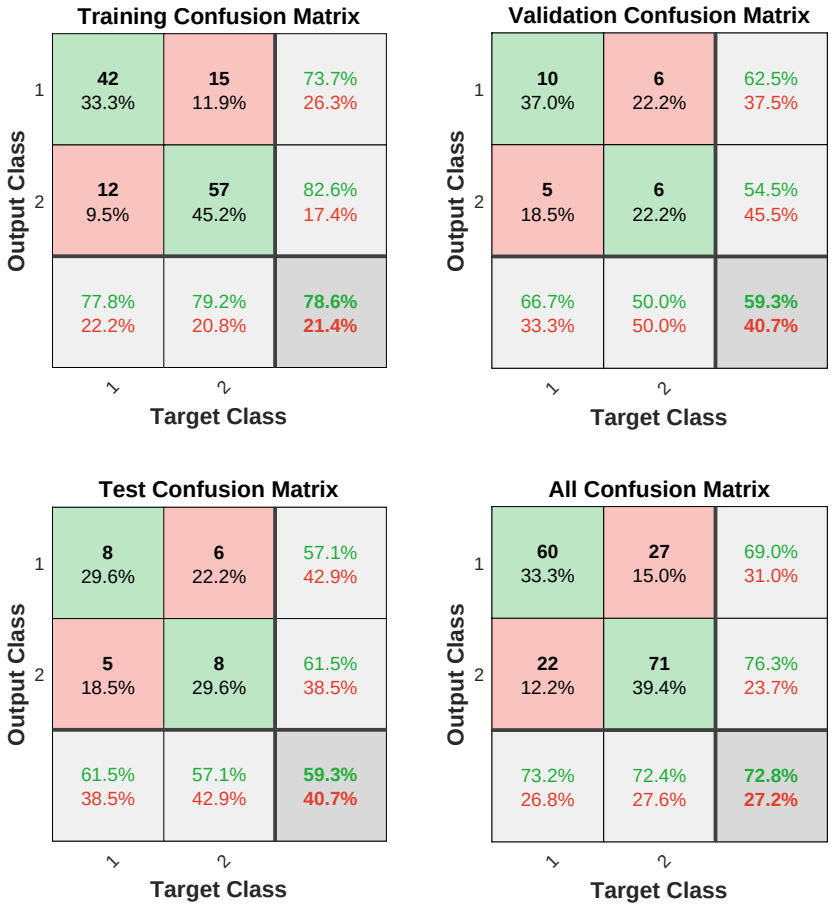


FIG. 3.10 – MLP performance of Aneurysm data.

Then the FI metric and the pairwise distances were computed as described in methodology section 3.2, followed by the FI pairwise distances that defined the Fisher manifold, leading to the similarity network incorporating an appropriate length scale σ_G for the Gaussian kernel.

The following figures (3.12 to 3.15) analyse the faithfulness of the network predictions as a function of the network length scale, σ_G. The σ_G parameter determines the size of the node neighbourhood, affecting the granularity with which the Newman's algorithm detects communities. At least initially, it is intended to find more communities than class labels, two in this case. The methodology applied is described in section 3.2.3.1, using the following indicators: KL-divergence, accuracy, Cramer's V and McNemar test.

In the figures, the blue lines represent the whole network, a weighted average of the MLP scores over all network elements, see eq. 3.19. The red lines represent the community predictions, an average of the adjacency matrix elements that belong to the same community. And the black lines represent the prototype predictions, the average of the MLP scores for the prototypes. The decision criteria will be based on the network behaviour (blue line).

Figure 3.11 shows the number of communities with respect to the network length scale, σ_G. The length scale is linearly sampled from the values comprised between 5% to 100% quantiles of the pairwise distances. From the figure one may observe that for σ_G greater than 1.2 the Newman's algorithm only find two communities in the created network, therefore $\sigma_G \leq 1.2$ is a threshold for having networks with the ability to discriminate more than two communities, which was the initial intention.

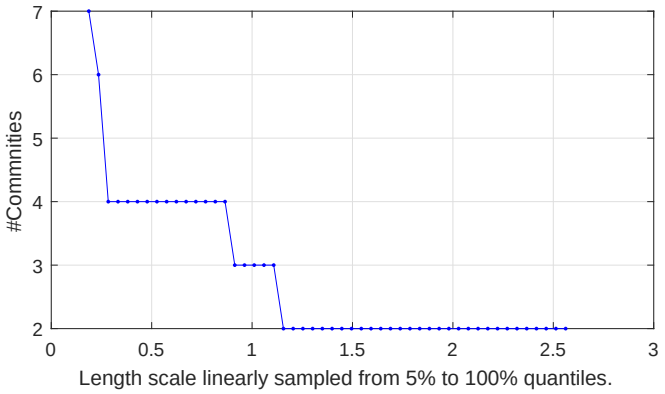


FIG. 3.11 – Aneurysm number of communities per σ_G

The figure 3.12 analyses the KL-divergence of the model based on the network predictions compared with the MLP predictions. This is the most important criterion because it is usually the most restrictive to select σ_G. The figure illustrates that the length scale should be $\sigma_G \leq 0.5$ to keep the KL-divergence of the network (blue line) close to zero.

The figure 3.13 shows the accuracy in the network predictions, considering all the nodes (blue line), community nodes (red line) and community prototypes (black). According to the network predictions (blue), the length scale should be $\sigma_G \leq 1.0$ to keep the accuracy on the same levels of the MLP.

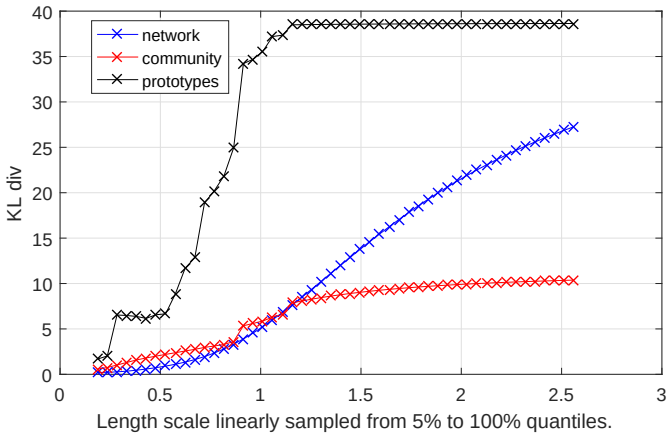


FIG. 3.12 – Aneurysm KL-divergence per σ_G

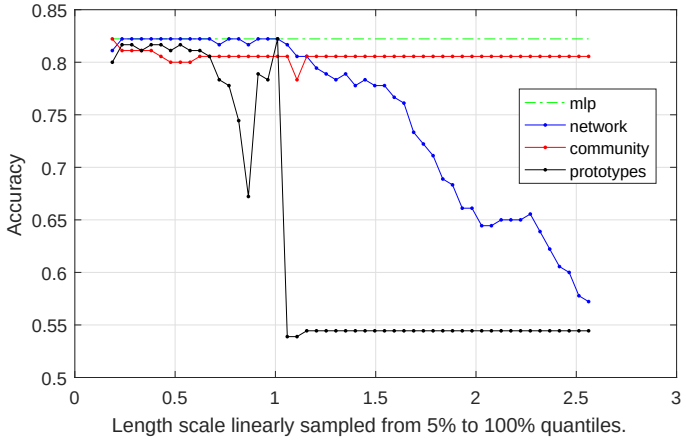


FIG. 3.13 – Aneurysm network predictions accuracy per σ_G

The figure 3.14 shows the Cramers' V statistic for the community labels compared with the class labels. This statistic measures the matching level between categorical variables. For this case this statistic is not very informative because most of the values are similar, approximately $\sigma_G \leq 0.75$ is a good threshold to keep $C_V \geq 0.64$ but it is not very relevant.

The figure 3.15 shows the McNemar tests, which is a measure of how different are the predictions of two models based on contingency tables, due to the null hypothesis is that both models are equal, big p-values indicate that the null hypothesis cannot be rejected. For the network model (blue) an acceptable length scale is $\sigma_G \leq 1.25$.

The idea was to select the highest length scale possible, preserving the faithfulness of the network predictions (blue lines). Usually the KL divergence is the most conservative indicator, figure 3.12 showing $\sigma_G \leq 0.5$ to avoid an increase in KL divergence. Given that the other plots point to a higher threshold, the decision is to keep $\sigma \approx 0.5$. If the second heuristic method was used based on intra-label manifold distances (see

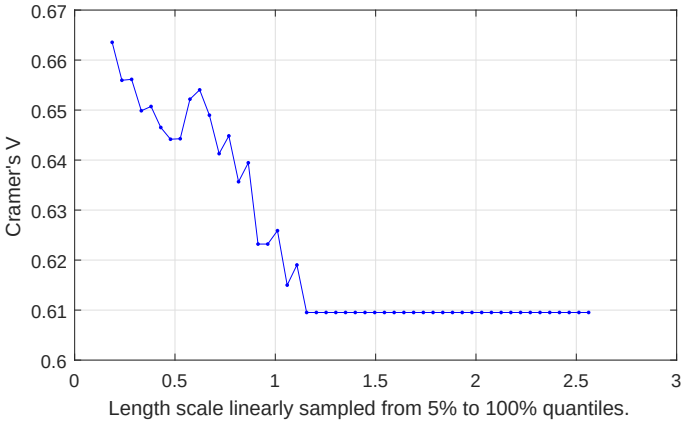


FIG. 3.14 – Aneurysm Cramer's V per σ_G

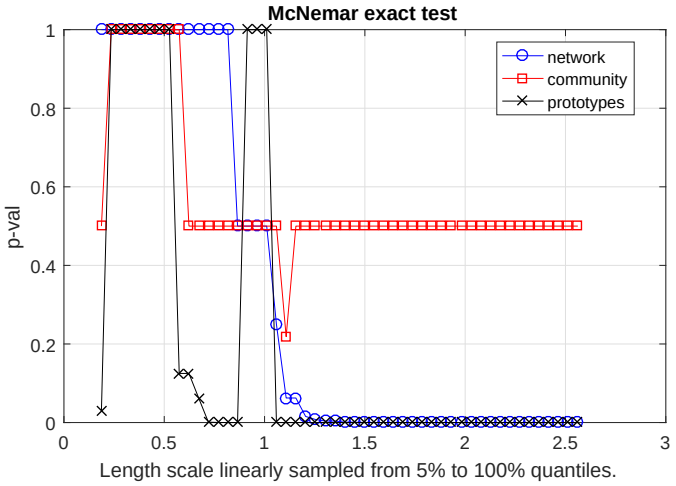


FIG. 3.15 – Aneurysm McNemar test per σ_G

Subsection 3.2.3.2), the value of σ would be 0.40, which indicated that both methods are broadly in agreement. For simplicity, when comparing the Fisher manifold with the manifold created by Euclidean pairwise distances, the heuristic method will be used to estimate the length scale.

Once the σ_G and the communities are found, the Fisher manifold was embedded and the communities represented.

Sammon mapping

Using the Sammon mapping, figures 3.16 and 3.17 show the differences between using the normal Euclidean distances to create the pairwise distance matrix, and the distances created by the Fisher manifold. For the Euclidean case there are five communities with significant overlapping, however for the Fisher case there are four well separated communities, displacing the ruptured aneurysm to the bottom right, and the not-ruptured to the left top of the figure.

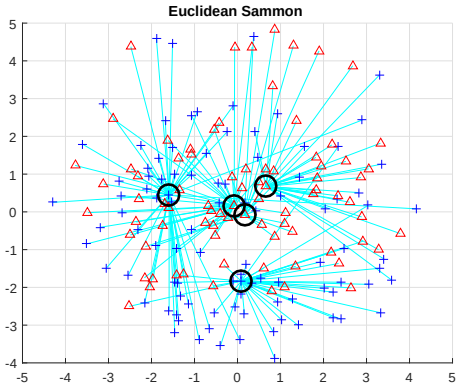


FIG. 3.16 – Aneurysm Sammon mapping of Euclidean dist.

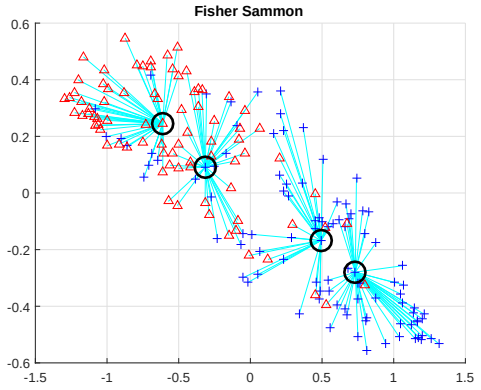


FIG. 3.17 – Aneurysm Sammon mapping of Fisher manifold

Classical MDS

Now using the cMDS with the 3 main eigenvectors there is a clear difference in the manifold structure between the Euclidean distances and the Fisher manifold, shown in figures 3.18 and 3.19.

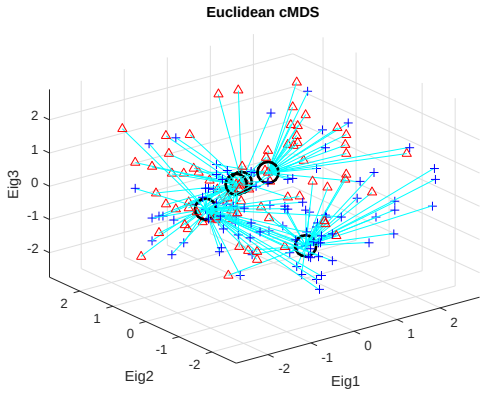


FIG. 3.18 – Aneurysm Euclidean dist. cMDS

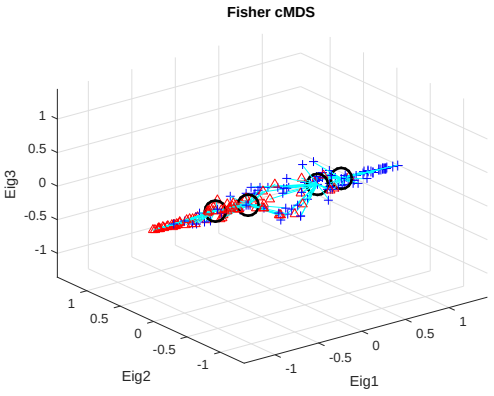


FIG. 3.19 – Aneurysm FIN cMDS in 3D

With figure 3.20 showing more information per community. For instance, the rupture prevalence per community, i.e. for each community, the ratio of the largest label against the community size (in communities Com2 and Com3 the largest label is not-rupture). This provides an indication of the expected prevalence of rupture within each community, therefore assigning patients to communities allows for an estimate of likelihood of rupture. In this analysis there are four communities, where the extreme right (Com1) has a probability of rupture of 94%. And the extreme left (Com2) has a probability of not-rupture of 82%. The intermediate communities form a more mixed group in terms of rupture where a decision on likelihood of rupture has to be taken more carefully, even having high prevalences (72% not-rupture in Com3 and 81% rupture in Com4).

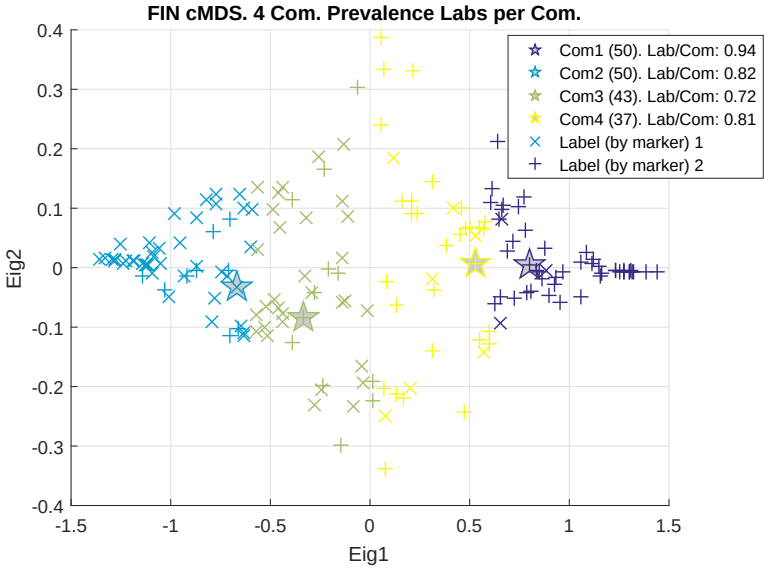


FIG. 3.20 – Aneurysm FIN cMDS with community information

Representing the eigenvalues of the cMDS in figure 3.21, one can observe that the Fisher manifold is practically one-dimensional. This happens in most of the cases with binary linear classifiers. For these cases only histograms are required of each class label across the main eigenvector, projecting all the manifold in this direction. Figure 3.22 shows the probabilities derived from the histogram, $P(X,C) = P(X|C) \cdot P(C)$, where $P(X|C)$ is the density histogram, and $P(C)$ is the class prevalence. The label-1 means not-ruptured aneurysm and label-2 ruptured.

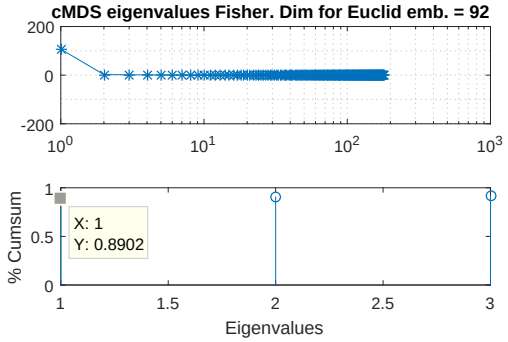


FIG. 3.21 – Aneurysm FIN cMDS eigenvalues

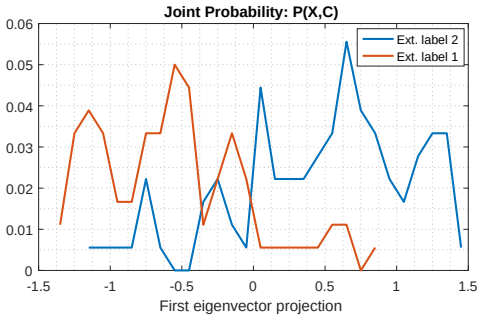


FIG. 3.22 – Aneurysm FIN cMDS histogram probabilities

The FIN procedure has been applied to the aneurysm dataset for representing a Fisher manifold and the communities defined by a similarity network. The objective was to obtain a rupture risk stratification beyond that of a binary classifier, and the analysis provided four communities, where the two extreme manifold communities have a rupture risk probability higher than the sensitivity and specificity of a binary classifier. Additionally, the histogram of the main eigenvector projection in the embedded Fisher

manifold has been constructed to produce a continuous probability distribution where the extremes belong to pure regions of 100% no-rupture (left) and 100% rupture (right) in figure 3.22. This is potentially very useful as it allows for a patient to be identified in the manifold, and evaluate the risk of rupture with respect to the other patients like them identified as their nearest neighbour.

3.4.2 Spirals

3.4.2.1 Data description

This is a synthetic dataset based on five nested spirals with Gaussian noise. There are 4K observations divided evenly into five classes, shown in figure 3.24. The spirals have little noise and are well separated.

Here the objective is to illustrate how a highly structured low-dimensional input space requires a non-linear classifier to produce a Fisher manifold of higher dimensionality. In most cases, the opposite usually happens, where the Fisher manifold reduces the dimensionality because it only considers those features that discriminate the classes.

3.4.2.2 Results and discussion

Applying the proposed hybrid solution, the MLP neuron weights are obtained from the Matlab implementation. The data is divided in 70% training, 15% validation and 15% test. The MLP results are shown in figure 3.23, the performance has been purposely made to be very good because the intention was to create an non-linear classifier with well separated class labels.

The Fisher pairwise distances with the straight-line approach are obtained in Spark, the rest of pipeline is performed in Matlab again, i.e. the shortest paths, the community detection and the Euclidean embedding.

If the Sammon mapping is applied to the spirals' Fisher manifold, figure 3.25, the Sammon mapping cannot separate the highly structure manifold in a 2D space. In fact, most of the communities do not overlap but this cannot be observed from the Sammon map. Therefore, Sammon mapping is not suited for representing non-linear, highly structured, interspersed Riemannian spaces.

Applying the cMDS on the spirals' Fisher manifold, its structure can be better appreciated. The method can approximate the embedding with 3 eigenvectors. Figure 3.26

FIG. 3.23 – MLP performance of 5 spirals data.

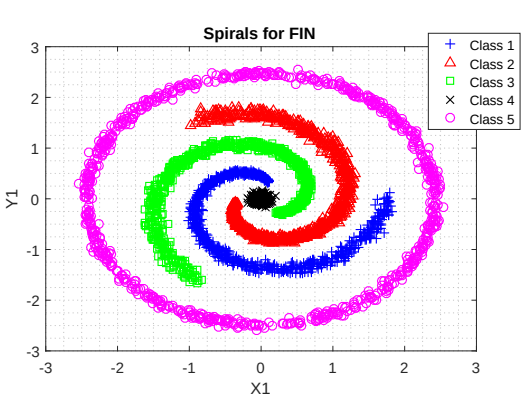


FIG. 3.24 – Spirals for FIN

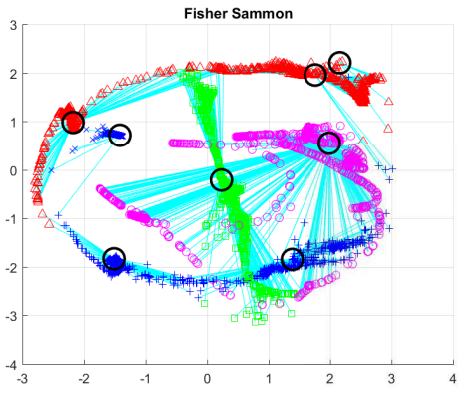


FIG. 3.25 – Spirals FIN with Sammon mapping

shows the first three eigenvalues that represent 81% of the cumulated variance. Although not apparent in the figure, the last eigenvalue has a significant negative value, $\lambda_{last} = -1513$ compared with the first eigenvalue $\lambda_1 = 15340$, which means the Riemannian manifold is considerably far from the expected behaviour of a Euclidean space.

Figure 3.27 illustrates the cMDS Euclidean embedding in 3D where one may observe that the Fisher manifold looks like the original spirals but distorted and stretched in the rotation axis direction. The Fisher manifold tries to improve the distinction between observations in different classes.

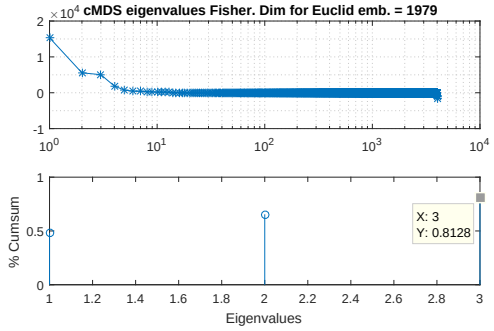


FIG. 3.26 – Spirals FIN cMDS eigenvalues

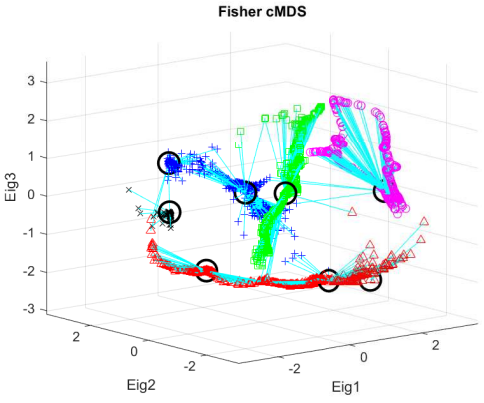


FIG. 3.27 – Spirals FIN cMDS 3D

In figure 3.28, the same plot is presented from another angle. The inner class 4 (black) is pushed to the left of Eig1 axis and the exterior class 5 (magenta) is pushed to the right of Eig1, in such a way that the 2D original space is transformed to a 3D space where the classes are completely separated.

Finally, figure 3.29 simply shows the cMDS built from Euclidean distances, because the original distances are Euclidean, the cMDS is able to reproduce the original distribution. However, the communities found in the Euclidean manifold mix the class labels.

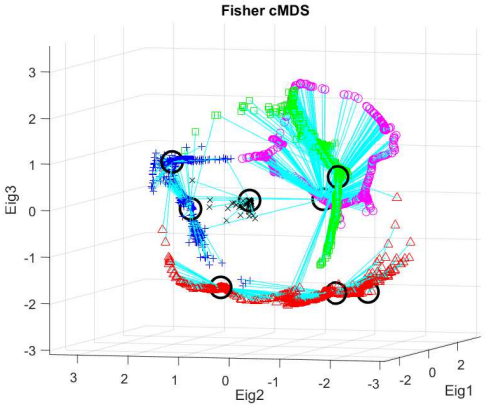


FIG. 3.28 – Spirals FIN cMDS 3D other angle

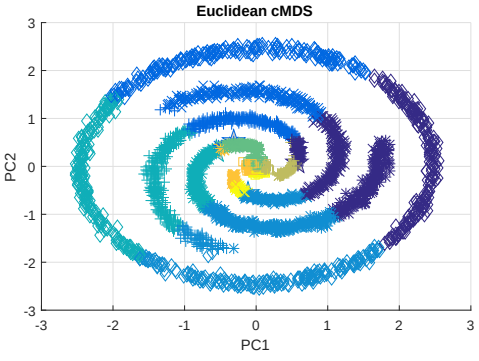


FIG. 3.29 – Spirals Euc. dist. cMDS mixed communities

3.4.3 Physics particle detection

3.4.3.1 Data description

The HEPMASS dataset is from the UCI repository [114]. It represents high-energy physics experiments focused on finding exotic particle signatures in the collisions. The data is obtained from Monte Carlo simulations of the signatures that produce the exotic particles and the resulting decay products. There is a target class label that separates the observations where an exotic particle is created from other observations with background sources. The dataset contains 27 normalized features and an additional extra column with the particle mass in case of particle creation. A random sample of 10K observations was taken from the original 10.5M instances.

The objective for this study is to use the FIN work-flow to classify the signatures that create the particles, firstly to find communities that discriminate the classifications, and secondly to inspect the structure of the Fisher manifold to gain insights about the class distributions.

3.4.3.2 Results and discussion

The Spark MLP obtains the performance depicted in figure 3.30, where the data has been divided in 70% training, 15% validation and 15% test.

The resulting Fisher manifold is firstly represented by Sammon mapping in figure 3.31, where the red points are no particle observations and the blue points the particles. Newman's algorithm finds four communities with the length scale obtained with the heuristic method. Figure 3.32 shows the maximum ratio of label membership per community, where the extreme communities identify observations with a high prevalence of particles, higher than the MLP accuracy. Community-1 has 87% of signatures of no-particles, while community-2 has 94% of signatures of finding of particles. The other two intermediate communities are a transition between the extremes.

If the particle masses associated are represented with each signature as in figure 3.33, it would show how the mass is directly correlated to the chance of finding a particle, with the signature of heavier particle easier to identify (right side of Sammon mapping).

For this dataset, the Sammon mapping has been useful for visualizing the distribution of the communities, but to observe the manifold structure itself, it is better to use the cMDS embedding. Figure 3.34 shows the eigenvalues of the cMDS, clearly representing one dominant eigenvector that represents 98% cumulative sum of the eigenvalues.

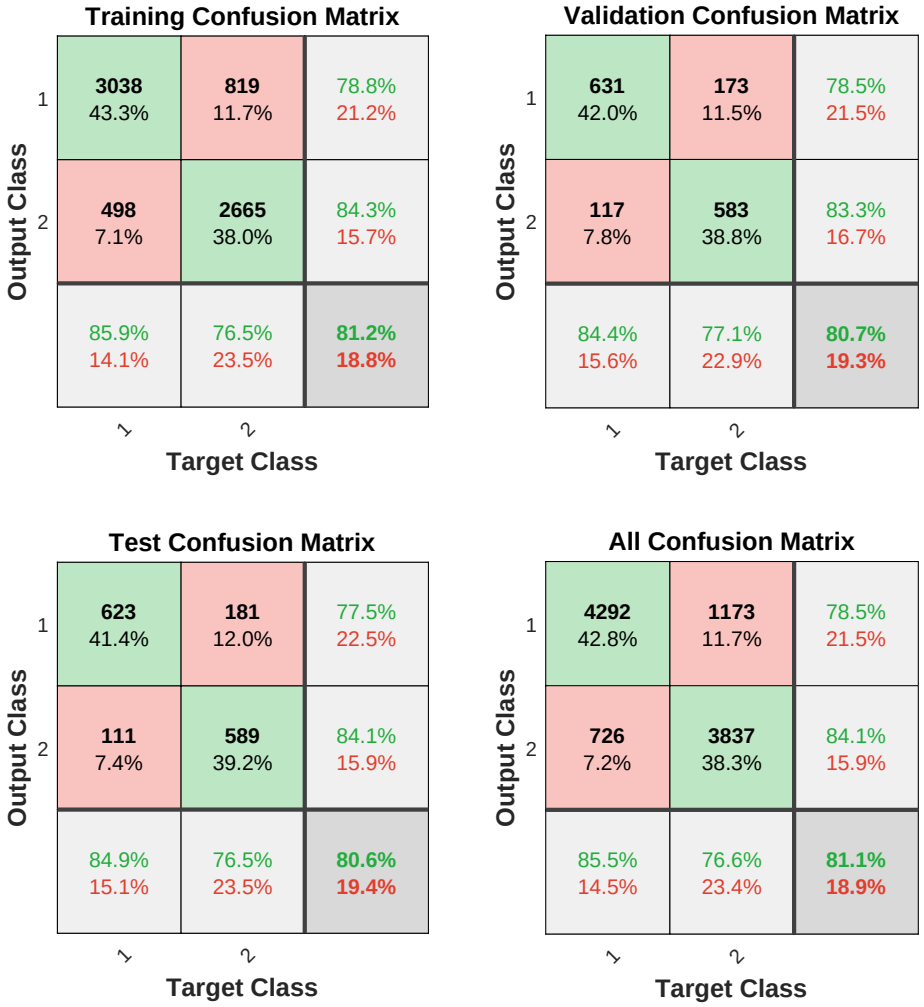


FIG. 3.30 – MLP performance of physics particles data.

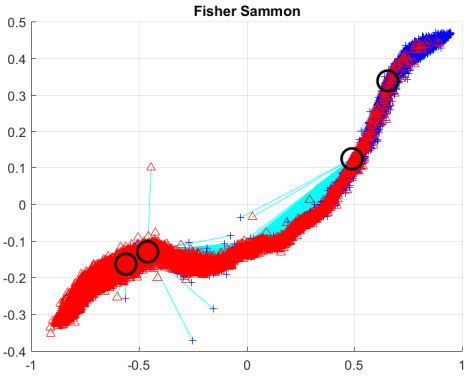


FIG. 3.31 – Particles FIN Sammon mapping

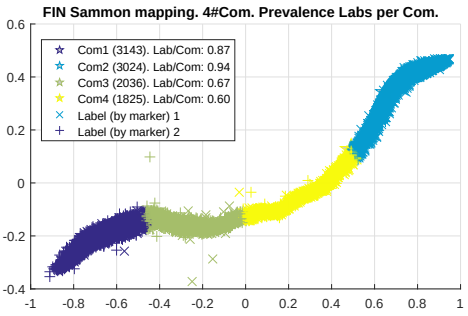


FIG. 3.32 – Particles FIN Sammon mapping by communities

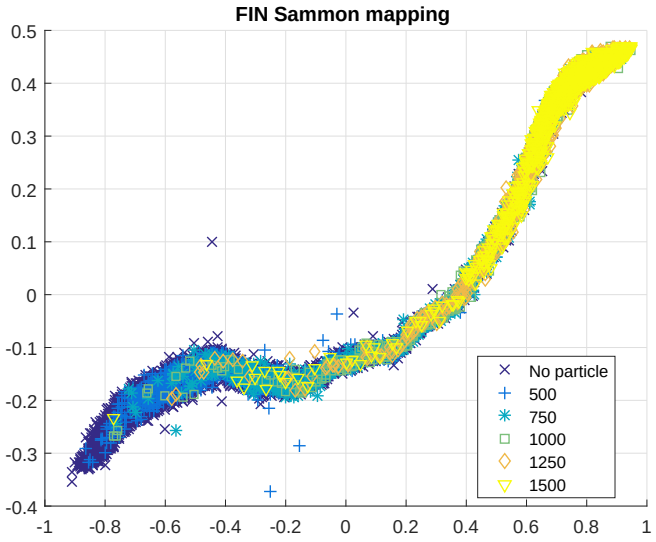


FIG. 3.33 – Particles FIN Sammon mapping by mass label

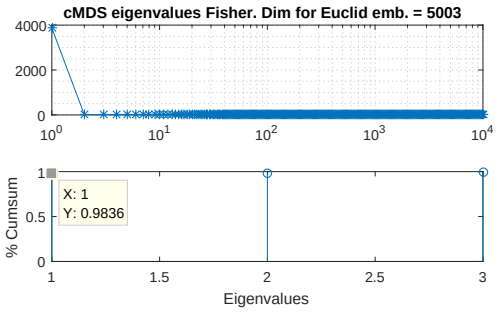


FIG. 3.34 – Particles FIN cMDS eigenvalues

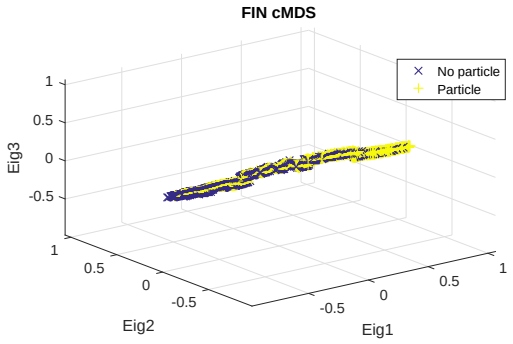


FIG. 3.35 – Particles FIN cMDS

The cigar shape of the manifold structure allows for an arbitrary segmentation, however given nearly all the observations lie in the first eigenvector, the manifold can be projected in this direction and histograms used to inspect the class distributions. Figure 3.36 shows that the manifold has two peaks at the extremes, the distribution could be a good criteria for segmenting cigar shape manifolds. If instead of using the class label the mass label is used, figure 3.37 shows how the particles mass are distributed in the Fisher manifold, highlighting the same effect observed in the previous mass Sammon map but with more detail and the mass distributions clearly identified.

Finally, a comparison of the cMDS created from Euclidean distances instead of the Fisher distances is presented. Recalling that cMDS obtains similar results to the PCA decomposition if the original distances were Euclidean, figure 3.38 shows the eigenvalue cumulative sums, where eigenvalue weights are distributed more evenly. This highlights the ability of the Fisher manifold to concentrate all the relevant information in one eigenvector, when the original data needs at least 15 eigenvectors to obtain an 80% of

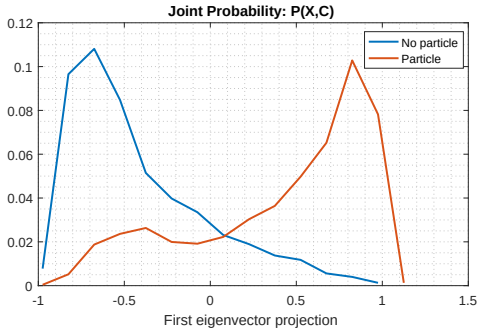


FIG. 3.36 – Particles FIN cMDS eig1 projection by class

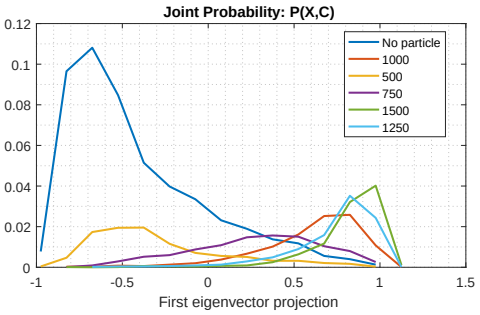


FIG. 3.37 – Particles FIN cMDS eig1 projection by mass

the total variance. However, plotting the two main eigenvectors distinguished by mass label 3.39, one may also observe that the information of mass distribution is mainly contained in the two main eigenvectors of the Euclidean distances manifold, analogously to PCA. Therefore, the particles and their mass can be discriminated with only the first principal component.

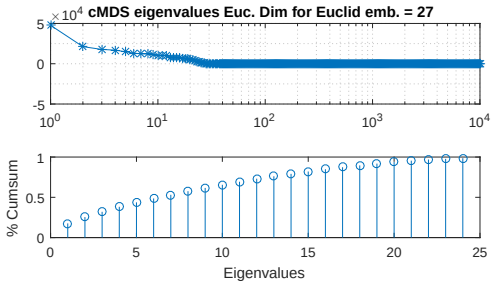


FIG. 3.38 – Particles Euclidean cMDS eigenvalues

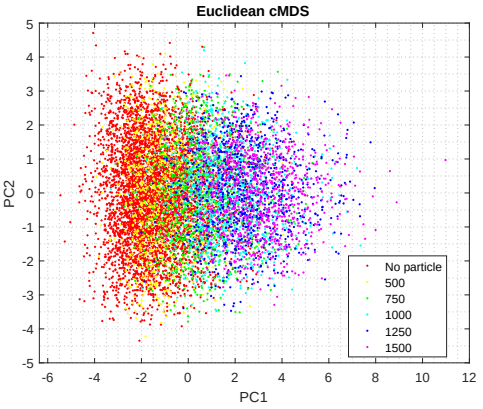


FIG. 3.39 – Particles Euclidean cMDS (PCA)

3.5 Conclusion

The core of the chapter is the manifold defined by the Fisher Information metric. This manifold contains the relevant information of the discriminative model which it is based on. Once the Fisher manifold is embedded in a Euclidean space and transforms the input space in a low-dimensional space, using the Sammon mapping it is helpful to visualize the communities created by the Newman's community finding algorithm. Alternatively, using the classical MDS the manifold structure can be visualized thanks to its eigenvector decomposition.

Reflecting on the results and the behaviour of the Fisher manifold, it tends to separate those observations that belong to the same class label, placing at the manifold extremes the observations with highest certainty of class membership, then moving towards the centre of the manifold there is a smooth transition from one class to another. This effect is related to the Fisher metric which depends on the gradient of the posterior class probabilities, the metric shortens the distances within regions of no class membership changes, and lengthens the distances within regions of class membership changes. The pairwise distances create a Riemannian manifold of minimum dimensionality that discriminates the class labels, because the Fisher metric only considers those features that contribute to differentiating the labels. That is the reason why in most of the cases where there is a binary classifier model, the manifold practically lies in one-dimension cigar shape, reflected in the Euclidean embedding with cMDS. For these cases the manifold structure is a continuous, homogeneous distribution where there are no truly separate clusters, it is more a segmentation problem where the distribution is split into an arbitrary number of communities, depending on the desired granularity. In general, the dimensionality of the Fisher manifold depends on the complexity of the classifier model, needing only one dimension to represent the linear binary classifiers. However, if the classifier model is multinomial or highly non-linear, the Fisher manifold will have a more complex structure producing an embedding with higher dimensionality.

In the first part of this chapter, the FIN pipeline has been successfully applied using small datasets, namely, the Iris and the aneurysm datasets. Given that the main limitation of this set of algorithms is the sample size when computing the Fisher metric and the pairwise distances, the second part of the chapter introduces some big data tools to tackle a greater number of observations in a scalable implementation of the FIN procedure.

In the second part of this chapter, a pseudo scalable version of the FIN pipeline has been implemented in Spark under Hadoop ecosystem, it is pseudo scalable because the quadratic dependence on the sample size with the pairwise distances will eventually lead to excessive runtimes. Two main bottlenecks have been identified: $Bn1$ and $Bn2$. $Bn1$, which is based on the pairwise distance computation with the straight-line approach (SL), can be sped up considerably using the Spark parallelization. $Bn2$, which is based on the computation of shortest paths across the Fisher manifold for approximating geodesic distances, is better tackled with the Floyd-Warshall algorithm in Matlab using a single machine (provided the data fits into memory). Therefore, the fastest option is a hybrid solution, tackling $Bn1$ with distributed computing (Spark) and $Bn2$ in a single machine (Matlab).

The criteria of the hybrid approach is the adopted solution from the perspective of

speeding up the runtime of research experiments, where only the fastest method combination is considered. However, hybrid approach is not elegant and the mixture of two environments may be not suitable for a hypothetical deployment in production.

In reference to the experiments, three case studies have been analysed, two of them with the scalable implementation. For the Aneurysm case and the exotic particles, a high dimensional input space is transformed into a one-dimensional Fisher manifold, where the class labels can be analysed using a density histogram. For the five nested spirals, because of the highly non-linear classifier model, the two-dimensional input space is transformed into a Fisher manifold of higher dimensionality, three-dimensional Euclidean embedding was used in this case.

As future work, without having the Hadoop ecosystem constraint of the retail company, it would be interesting to address the problem with GPGPUs, where the level of parallelization is very high and the whole pipeline could be implemented in the same machine.

Chapter 4

Probabilistic Quantum Clustering

This chapter shows in detail the new methodology of probabilistic Quantum Clustering and how it is derived from the original QC. This density-based clustering algorithm is useful to find clusters in the embedded manifold when the space presents high density variations.

One of the main advantages of PQC is its ability to automatically select the hyper-parameters though a likelihood score of cluster membership. The Subsection 4.2.5 shows that this likelihood function, based on $P(K|\mathbf{X})$, is correlated with the Jaccard score (JS) when the underlying cluster structure is known for synthetic data.

The rest of the chapter is structured as follows: Section 4.2 introduces the original Quantum Clustering, including a description of the algorithm in Subsection 4.2.1, then the proposed QC improvements based on KNNs are described in Subsection 4.2.2, the manifold QC in 4.2.3, the probabilistic interpretation of the QC in 4.2.4; Subsection 4.2.5 shows how the likelihood function becomes an effective way of assessing the performance of QC, and subsections 4.2.6 and 4.2.7 show auxiliary methods to identify potential wells with clusters. Section 4.3 presents the data sets used to test the performance of the proposed clustering method with results reported in Section 4.4. Section 4.5 concludes with a critical summary of PQC identifying directions for further work.

4.1 Introduction

In the FIN procedure, once the Fisher manifold embedded in a Euclidean space with the cMDS is obtained, communities or clusters can be found directly on the new Euclidean space using traditional projective methods like K-Means, DBSCAN [24], mixture of Gaussians or any other similar clustering algorithm; instead of the spectral methods used

previously based on pairwise similarity matrices. One of the disadvantages of spectral clustering is the need for setting up an additional hyper-parameter that defines the length scale in the Gaussian kernel for controlling the influence area of the neighbourhood similarity. Most of the algorithms based on projective methods also need additional hyper-parameters, like the cluster number K in K-Means, or in the mixture of Gaussians before the Expectation-Maximization (EM) fitting. On the other hand, density-based algorithms like DBSCAN can detect automatically the cluster number, but they still need a hyper-parameter for controlling the neighbourhood length scale.

Part of the work in this thesis has focused on finding an unsupervised clustering algorithm able to detect the data structure assessing the hyper-parameter selection. This chapter introduces such an algorithm which is a new probabilistic framework developed for improving the original Quantum Clustering [69]. The new Probabilistic Quantum Clustering (PQC) introduces many new improvements that will be explained in detail in this chapter, which include the ability to determine the cluster number and the data density structure in a completely unsupervised way.

Since the PQC is a density-based algorithm, it fits very well in the FIN pipeline when the embedded Fisher manifold presents a data structure with local density variations, i.e. heteroscedastic data. However, it is recommended to avoid the use of PQC in cases where the Fisher manifold presents a homogeneous distribution with a constant density, as in this situation the community finding step becomes a segmentation problem where the cluster number is quite arbitrary. For instance, in a uniform cigar-shape dataset, PQC tends to identify a single big cluster without segmenting the data.

4.2 Methodology

This Section starts (Subsection 4.2.1) with a general description of the original QC and its main elements, namely, the wave function, the potential and the potential gradient. Then, follows the introduction of the new approaches alongside its implications culminating in a probabilistic model with an unsupervised assessment for the free hyper-parameter.

In case of heterogeneous features, the data are sphered by standardizing each dimension to the z-score, in order to provide a uniform length scale across all dimensions. Additionally, the data is scaled by a constant, $1/\lambda$, to make the length scale uniform when the stochastic gradient descent (SGD) is applied:

$$\lambda = \frac{\sum_{i=1}^{n} \|\mathbf{x}_i\|}{n} \tag{4.1}$$

where n is the length of vector $\mathbf{x}$, and $\|\mathbf{x}\|$ is the L^2 norm.

4.2.1 Original Quantum Clustering, QC_σ

QC_σ starts by defining a wave function as a Parzen estimator, from which a convex potential function is derived by the Schrödinger equation. Cluster allocation consists in identifying which regions belong to each local minimum of the potential function, originally through a gradient descent.

Gaussian kernels associated with each observation add together to make the wave function (4.2):

$$\Psi(\mathbf{x}) = \sum_{i=1}^{n} \psi_i(\mathbf{x}) = \sum_{i=1}^{n} e^{-\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2\sigma^2}} \tag{4.2}$$

where n is the sample size and σ a global length scale comprising a single hyper-parameter to adjust. The Gaussian normalisation $(\sqrt{2\pi}\sigma)^{-d}$ is redundant as it will cancel out in the calculation of the potential function. Applying the eq. 2.20 the potential is:

$$V(\mathbf{x}) = E + \frac{\sigma^2}{2}\frac{\nabla^2\Psi(\mathbf{x})}{\Psi(\mathbf{x})} = E - \frac{d}{2} + \frac{\sum_{i=1}^{n}(\mathbf{x}-\mathbf{x}_i)^2 e^{-\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2\sigma^2}}}{2\sigma^2\Psi(\mathbf{x})} \tag{4.3}$$

where d is the dimension of the input space, and E in this context plays the role of an offset of $V(\mathbf{x})$. If $V(\mathbf{x}) \geq 0$ is imposed, then $E = -min\frac{\sigma^2}{2}\frac{\nabla^2\Psi(\mathbf{x})}{\Psi(\mathbf{x})}$. For the purposes of cluster allocation with the gradient descent, the offset values of $V(\mathbf{x})$ are irrelevant.

Therefore, QC_σ potential corresponds to a weighted average of the function F over Ψ, and it could be expressed as the expected value of F over Ψ, as follows:

$$\langle F_i \rangle_\Psi \equiv \frac{\sum_i F_i\psi_i}{\sum_i \psi_i} \tag{4.4}$$

Applying the gradient to $\langle F \rangle_\Psi$ yields a generic expression useful for simplifying the QC_σ equations:

$$\nabla \langle F_i \rangle_\Psi = \left\langle \nabla F_i - \frac{F_i}{\sigma^2} (\mathbf{x} - \mathbf{x}_i) \right\rangle_\Psi + \left\langle F_i \right\rangle_\Psi \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)}{\sigma^2} \right\rangle_\Psi \tag{4.5}$$

With this notation the $V(\mathbf{x})$ and $\nabla V(\mathbf{x})$ are simplified to:

$$V(\mathbf{x}) = E + \frac{\sigma^2}{2} \frac{\nabla^2 \Psi(\mathbf{x})}{\Psi(\mathbf{x})} = E - \frac{d}{2} + \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma^2} \right\rangle_\Psi \tag{4.6}$$

$$\nabla V(\mathbf{x}) = \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)}{\sigma^2} \right\rangle_\Psi \left(1 + \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma^2} \right\rangle_\Psi \right) - \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)\,(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma^4} \right\rangle_\Psi \tag{4.7}$$

The next step in the QC_σ is to apply the gradient descent to allocate the clusters. Defining $\mathbf{y}_i(0) = \mathbf{x}_i$, the usual gradient descent is:

$$\mathbf{y}_i(t + \Delta t) = \mathbf{y}_i(t) - \eta(t) \nabla V(\mathbf{y}_i(t)) \tag{4.8}$$

where $\eta(t)$ is the learning rate.

A variant of SGD is applied with an adaptive momentum term ADAM [115] which makes it suitable for sparse gradients that commonly occur with sparse data or outliers. In order to ensure the convergence of SGD, two criteria are imposed:

$$max(|\Delta \mathbf{y}_i|) \leq \epsilon_y \qquad max(\Delta V(\mathbf{y}_i)) \leq \epsilon_V \tag{4.9}$$

where ϵ is the threshold[1]. The first stopping criterion ensures that the updating distances in SGD are smaller than a given threshold, while the second limits the size of potential differences. The next step is to identify the clusters allocated to particular local minima of the potential function. This is detailed in Section 4.2.6.

One of the main limitations of QC_σ is having a single length scale which sets equal width for all Gaussian functions irrespective of local density. Length scales ranging from the

[1]Empirically, a good value for both thresholds is $\epsilon \approx 0.001$ if the data have been rescaled to have an average length of 1.

smallest to the average pairwise distance between observations will in general detect a cluster per observation or a single cluster for all of the data. A parametrisation of the length scale that reflects variations in local density is the average of pairwise distances between observations ordered by proximity i.e. nearest neighbours, as shown below:

$$\sigma_{k\%} = \frac{1}{n}\sum_{i}^{n}\sum_{j\,\in\,knn} dist(\mathbf{x}_j, \mathbf{x}_i) \tag{4.10}$$

In order to compare different QC methodologies an artificial data set #1 is introduced, which is further detailed in Section 4.3.1. It consists of four two-dimensional clusters some of which are strongly anisotropic, as well as a high-density cluster nested within a low-density one. Each cluster has 100 observations.

Figure 4.1 shows the cluster allocation by SGD from the potential gradient of the QC_σ model with a length scale of $\sigma_{20\%}$ showing the corresponding direction of the gradient vectors in figure 4.2. The length scale adjusted for the all of the data is too broad to accurately capture the high density cluster and too narrow for the sparse cluster at the bottom of the plot which breaks up into multiple local minima. The resulting Jaccard score against the clusters identified by the generating density functions is 0.556.

Using QC_σ model with a length scale of $\sigma_{20\%}$, figure 4.1 shows the cluster allocation by SGD from the gradient of the potential, figure 4.2 depicts the direction of the potential gradient vectors.

The wave function, and consequently the potential, are too smooth to fit the local density changes, thus providing a biased clustering. This example is a straightforward example of the difficulties of QC_σ to classify data whose density is locally variable.

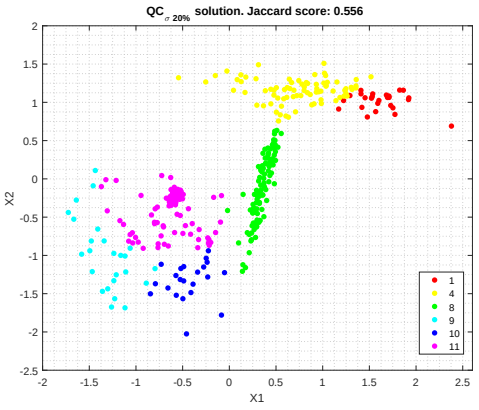


FIG. 4.1 – SGD alloc. $QC_{\sigma\,20\%}$

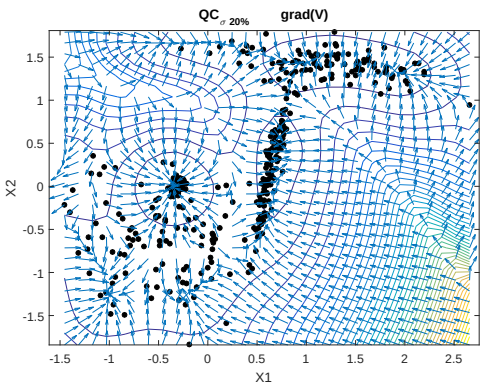


FIG. 4.2 – Gradient $QC_{\sigma\,20\%}$

The improvements in clustering shown in figure 4.3 show the potential of adjusting the local variance around data points. This approach has been explored in the literature [21, 72, 81]. However, there remains a need to find a more principled approach to resolve the cluster allocation problem for heteroscedastic data. This is addressed in the next Section.

4.2.2 K-neighbours Quantum Clustering, QC_{knn}

Information about local density can be included in the length scale by defining σ as a function of the KNNs, where the new hyper-parameter is the quantity of neighbours to consider. This quantity will be expressed as a percentage of the total sample size: $K = \%\text{KNN}$.

$$\sigma_i \equiv \frac{1}{K} \sum_{j \in knn(\mathbf{x}_i)}^{K} dist(\mathbf{x}_i, \mathbf{x}_j) \tag{4.11}$$

Each observation contributes a different Gaussian function to the overall wave function in eq. 4.12. Multiplying through by $\frac{1}{n}$ ensures correct normalisation of the integral over the input space, $\int_{\mathbb{R}} \Psi(\mathbf{x}) d\mathbf{x} = 1$.

$$\Psi(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \psi_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{e^{-\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2\sigma_i^2}}}{\left(\sqrt{2\pi}\sigma_i\right)^d} \tag{4.12}$$

where d is the dimensionality of the sample. One may observe that in (2.19) the total kinetic term is decoupled, $T(\mathbf{x}) = \frac{-\sigma^2}{2}\nabla^2\Psi(\mathbf{x})$ (σ^2 and ∇^2 are separated factors because σ is a constant common factor), now, with a different σ_i per observation, the Schrödinger eq. 2.19 has to be updated. The kinetic term of each observation, T_i, can be expressed as follows:

$$T_i = \frac{\sigma_i^2}{2}\nabla^2\psi_i = \left(\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2\sigma_i^2} - \frac{d}{2}\right)\psi_i \tag{4.13}$$

Therefore, the new total kinetic term couples the length scale σ_i and $\nabla^2\psi_i$:

$$T_{total} = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \frac{\sigma_i^2}{2} \nabla^2 \psi_i \tag{4.14}$$

The new potential and its gradient are similar to eq. 4.6 and eq. 4.7, but with a variable σ_i:

$$V(\mathbf{x}) = E + \frac{\sum_i \frac{\sigma_i^2}{2} \nabla^2 \psi_i}{\sum_i \psi_i} = E - \frac{d}{2} + \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma_i^2} \right\rangle_\Psi \tag{4.15}$$

$$\nabla V(\mathbf{x}) = \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)}{\sigma_i^2} \right\rangle_\Psi \left(1 + \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma_i^2} \right\rangle_\Psi \right) - \left\langle \frac{(\mathbf{x} - \mathbf{x}_i)\,(\mathbf{x} - \mathbf{x}_i)^2}{2\sigma_i^4} \right\rangle_\Psi \tag{4.16}$$

Using again the artificial data set #1 as an example, the variable length scale produces a wave function with a very pronounced peak in the high density region causing a "volcano effect" in the potential (figure 4.4). The shape of QC_{knn} potential is much more complex than that obtained by QC_σ, as it is now smooth in sparse regions and steep in dense areas, as required. Figure 4.3 shows the cluster allocation by SGD over this potential with accurate discrimination of the high density cluster against the surrounding sparse cluster. The potential also adapts to the local density changes, creating a sharp sink around the highest density peak; this region will be isolated in the clustering allocation by gradient descent (SGD), allowing a cluster discrimination by local densities. In this example, $\sigma_{20\%}$ is an appropriate parameter value, if σ were much smaller it would produce an overfitted potential, generating too many sub-clusters.

The Jaccard score in QC_{knn} (0.862) is much better than in QC_σ (0.556).

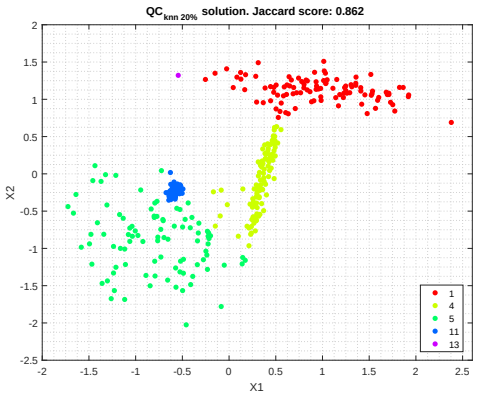


FIG. 4.3 – SGD alloc. $QC_{knn\,20\%}$

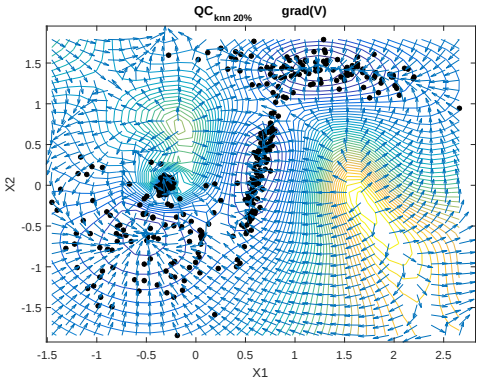


FIG. 4.4 – Gradient $QC_{knn\,20\%}$

Adjusting the length scale from nearest neighbours is clearly effective for detecting clusters with very different densities and also to accommodate outliers with smooth and flat

gradients that do not lead to unnecessary fragmentation in low density regions. This Section has proposed an approach that partially solves the problem of heteroscedasticy but the amount of neighbours considered in the model is still a hyper-parameter to be determined. There is a trade-off between too few neighbours resulting in an overfitted density function with too many clusters, and too large a neighbourhood leading to a biased density function with too few clusters.

4.2.3 Covariance-based Manifold Quantum Clustering, QC_{cov}

Previous work has proposed Gaussian kernels with non-spherical covariance matrices estimated from local manifold information [84]. The local covariance matrix, Σ_i is computed using the relative distribution of the KNNs around each observation:

$$\Sigma_i = \frac{1}{N_k - 1} \sum_{j \in knn}^{N_k} (\mathbf{x_j} - \mathbf{x}_i)^T (\mathbf{x_j} - \mathbf{x}_i) \tag{4.17}$$

Now, each observation has a kernel with the form of a multivariate normal distribution, producing the following wave function:

$$\Psi(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \psi_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{|2\pi\Sigma_i|}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T \Sigma_i^{-1} (\mathbf{x}-\mathbf{x}_i)} \tag{4.18}$$

This wave function is a more accurate probability density function than those presented in Sections 4.2.1 and 4.2.2 since each observation captures the distribution of the nearest neighbours. The density function reproduces faithfully elongated distributions, like cigar shapes or even spiral shapes, a case study in Section 4.4. But it also has some disadvantages:

1. The mathematical complexity of the equations for the potential function and corresponding gradients is significantly increased.

2. Degenerate covariance matrices i.e. with diagonal element close to zero, may cause singularities in the covariance inverse.

3. If the covariances are too anisotropic, the positive effect of superposition in the wave function is considerably reduced. This produces a wave function that is less smooth and a potential less convex, favouring the creation of an excess number of local minima if the Gaussian kernels do not overlap one another enough.

These disadvantages can be mitigated if all the local covariance matrices are restricted to be diagonal although anisotropic (expressed into their eigenvector basis). Degeneracy is avoided by setting a minimum threshold value for the diagonal elements, as shown in Section 4.2.7. This also improves the superposition effect in the wave function, because all Gaussian kernels have a minimum radius controlled by the local-covariance threshold, but a larger ellipsoid axis greater when necessary. The local-covariance threshold is defined by:

$$\sigma^2_{th_i} = \frac{\sigma^2_{k'nn_i}}{d} \tag{4.19}$$

where d is the dimension of the data and $\sigma_{k'nn_i}$ the mean distance of the k' nearest neighbours of each observation i, namely, the variable length scale used in $QC_{k'nn}$. The percentage of neighbours considered, k', is determined experimentally in Section 4.2.7; results show that k' should be the same k used to compute the local covariance matrix, in order to keep enough interaction between kernels; this means that the best QC model is a hybrid model between QC_{knn} and QC_{cov}.

The Schrödinger eq. 2.19 must be adapted to the variable length scale σ_i, as in eq. 4.13 when the kinetic term had coupled σ_i^2 and $\nabla^2\psi_i$. In this case, σ_i must be replaced by a scalar expression of Σ_i; considering that Σ_i is diagonal if it is expressed in an eigenvector basis, it is possible to make the change $\sigma_i^2 \rightarrow tr(\Sigma_i)$.

The derivation of $\nabla^2\psi_i$ is based on the equations proposed in [116]:

$$\frac{\partial\psi_i}{\partial\mathbf{x}} = -\psi_i\,\Sigma_i^{-1}\,(\mathbf{x}-\mathbf{x}_i) \tag{4.20}$$

$$\frac{\partial^2\psi_i}{\partial\mathbf{x}\,\partial\mathbf{x}^T} = -\psi_i\left(\Sigma_i^{-1}\,(\mathbf{x}-\mathbf{x}_i)\,(\mathbf{x}-\mathbf{x}_i)^T\,\Sigma_i^{-1} - \Sigma_i^{-1}\right) \tag{4.21}$$

$$\nabla^2\psi_i = tr\left(\frac{\partial^2\psi_i}{\partial\mathbf{x}\,\partial\mathbf{x}^T}\right) \tag{4.22}$$

The potential has the following expression:

$$V(\mathbf{x}) = E + \frac{\sum_i \frac{tr(\Sigma_i)}{2}\psi_i \; tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1} - \Sigma_i^{-1}\right)}{\sum_i \psi_i} = \\ = E + \left\langle \frac{tr(\Sigma_i)}{2} tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1}\right)\right\rangle_\Psi - \left\langle \frac{1}{2} tr\left(\Sigma_i\right) tr\left(\Sigma_i^{-1}\right)\right\rangle_\Psi \tag{4.23}$$

To obtain the gradient of V, the following expression is needed:

$$\frac{\partial \; tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1}\right)}{\partial \mathbf{x}} = 2\left(\Sigma_i^{-1}\right)^2\left(\mathbf{x}-\mathbf{x}_i\right) = 2\Sigma_i^{-2}\left(\mathbf{x}-\mathbf{x}_i\right) \tag{4.24}$$

The gradient of V can be split into two terms, $\nabla V = \nabla V_1 - \nabla V_2$

$$\begin{aligned} \nabla V_1 = \nabla\Big\langle \frac{tr(\Sigma_i)}{2} tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1}\right)\Big\rangle_\Psi = \\ +\Big\langle tr(\Sigma_i)\Sigma_i^{-2}\left(\mathbf{x}-\mathbf{x}_i\right)\Big\rangle_\Psi + \\ -\Big\langle \frac{tr(\Sigma_i)}{2} tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1}\right)\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\Big\rangle_\Psi + \\ +\Big\langle \frac{tr(\Sigma_i)}{2} tr\left(\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\left(\mathbf{x}-\mathbf{x}_i\right)^T\Sigma_i^{-1}\right)\Big\rangle_\Psi \Big\langle \Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\Big\rangle_\Psi \end{aligned} \tag{4.25}$$

$$\begin{aligned} \nabla V_2 = \nabla\Big\langle \frac{1}{2} tr\left(\Sigma_i\right) tr\left(\Sigma_i^{-1}\right)\Big\rangle_\Psi = \\ -\Big\langle \frac{tr(\Sigma_i)tr(\Sigma_i^{-1})}{2}\Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\Big\rangle_\Psi + \Big\langle \frac{tr(\Sigma_i)tr(\Sigma_i^{-1})}{2}\Big\rangle_\Psi \Big\langle \Sigma_i^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\Big\rangle_\Psi \end{aligned} \tag{4.26}$$

The sample local covariance estimate is diagonalized to threshold eigenvalues to a minimum value that is higher than a small regularization term, making further regularization unnecessary. This imposes a minimum radius in the covariance matrix resulting in QC_{cov} that are similar in value to QC_{knn}. In contrast, the gradient and equipotential surfaces of figure 4.6 show contours that better capture anisotropy. However, the QC_{cov} potential is less smooth than QC_{knn} potential so tends to create more local minima. These local minima create sub-clusters as seen in figure 4.5, which shows the allocation of seven clusters. The occurrence of these sub-clusters reduces performance (Jaccard score of 0.805) motivating the next Section.

4.2.4 Probabilistic Quantum Clustering, QC_{cov}^{prob}

Probabilistic Quantum Clustering QC_{cov}^{prob} modifies the cluster allocation of QC_{cov}. In particular, the clusters are no longer defined by the groups of points found after the

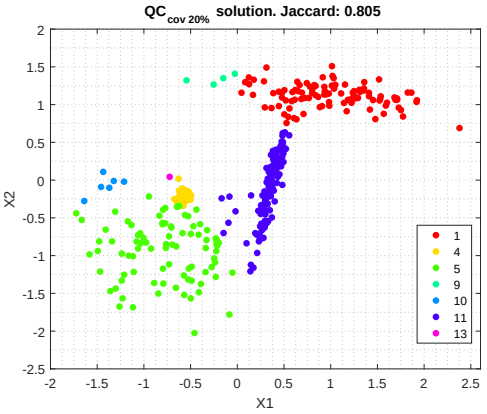


FIG. 4.5 – SGD alloc. $QC_{cov\,20\%}$

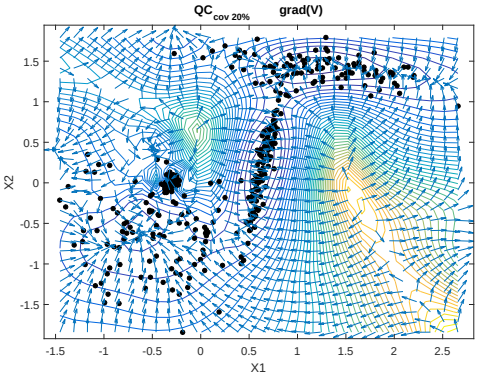


FIG. 4.6 – Gradient $QC_{cov\,20\%}$

SGD. These groups are now used to define component elements (subfunctions) that add to make the overall wave function.

The starting point for the probabilistic framework QC^{prob}_{cov} is to attribute the joint probability of observing cluster k in the position $\mathbf{x}$ to the sum of Gaussian functions associated with the observations grouped in the cluster (subfunction) k:

$$\Psi(\mathbf{x}) = \sum_{k=1}^{K} \frac{\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})}{n} = \sum_{k=1}^{K} P(k,\mathbf{x}) = P(\mathbf{x}) \tag{4.27}$$

where n is the sample size, K the total number of clusters, and $\#k$ the number of observations in cluster k.

Eq. 4.27 could be seen as a generative model using a mixture of Gaussians, one centred at each observation, with prior probability equal to $1/n$ (isotropic Gaussian kernel over the data). The purpose of the QC is to provide a link between the individual Gaussian functions so that points in the same cluster are linked together. This requires the use of the Schrödinger equation.

The probability of k can be obtained by marginalizing the joint probability over $\mathbb{R}$:

$$P(k) = \int_{\mathbb{R}} P(k,\mathbf{x})d\mathbf{x} = \int_{\mathbb{R}} \frac{\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})}{n} d\mathbf{x} = \sum_{i\in k}^{\#k} \frac{\int_{\mathbb{R}} \psi_i(\mathbf{x})\, d\mathbf{x}}{n} = \sum_{i\in k}^{\#k} \frac{1}{n} = \frac{\#k}{n} \tag{4.28}$$

Once the joint probability is defined, the Bayes' rule can be applied to obtain the following probabilities:

$$P(k|\mathbf{x}) = \frac{P(k,\mathbf{x})}{P(\mathbf{x})} = \frac{\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})}{\sum_{k=1}^{K}\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})} \tag{4.29}$$

$$P(\mathbf{x}|k) = \frac{P(k,\mathbf{x})}{P(k)} = \frac{\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})}{\frac{\#k}{n}} \tag{4.30}$$

Now, $P(k|\mathbf{x})$ can be used to define a new probabilistic cluster allocation:

$$cluster(\mathbf{x}) = \arg\max_k P(k|\mathbf{x}) \tag{4.31}$$

In other words, the cluster allocation in the region $\mathbf{x}$ will correspond to the cluster k such as $\arg\max_k P(k|\mathbf{x})$, or equivalently $\arg\max_k P(k,\mathbf{x})$ since $P(\mathbf{x})$ is a common denominator.

Summing up, there are two cluster allocations in the algorithm work-flow:

The first one is when the gradient descent is performed over the potential to allocate each observation of the training set into its corresponding potential well (identified as cluster). This gives the grouped Gaussians per cluster K; according eq. 4.28, the probabilistic framework can be derived: $P(k,\mathbf{x}) = \frac{1}{n}\sum_{i\in k}^{\#k} \psi_i(\mathbf{x})$

The second cluster allocation, called probabilistic cluster allocation, is based on the probabilistic framework, and it only decides to which cluster each observation belongs based on the $P(k|\mathbf{x})$, selecting k so that makes $P(k|\mathbf{x})$ maximum. The probabilistic cluster allocation can allocate any observation of the input space.

As a consequence of this, it is possible that the model generates k clusters but not all of them contain observations. In other words, there would exist k' empty (small) clusters if $P(k'|\mathbf{x})$ never wins in any region of the input space, $\mathbf{x}$.

This is a significant improvement on the original method for cluster allocation because any region of input space can be allocated to a cluster without the need to apply SGD over the potential. The probabilistic cluster allocation draws a probability map based on $P(k|\mathbf{x})$ to define the boundaries between clusters. However the probability map still requires one cluster allocation by SGD (Section 4.2.6 describes the procedure using a community detection after SGD). But for new observations no additional SGD is required.

Experimental results show a difference lower than 2% in the cluster allocation when comparing QC_{knn} with its probabilistic counterpart. Differences are greater in the case of QC_{cov}, with the probabilistic cluster allocation closer to the true labels than the SGD approach, with a Jaccard score of 0.882. The probabilistic approach improves the result by selecting only four of the seven clusters detected in figure 4.5. This is the model that provides the highest Jaccard score (JS) for data set #1 of all the tests carried out.

Another interesting characteristic of the probabilistic approach is the capability to implement outlier detection where the probability of belonging to any cluster is lower than a given threshold. For instance, setting 5% as the outlier threshold:

$$\textit{If in a region } \mathbf{x} \in \mathbf{X},\ P(\mathbf{x}|k) < 5\% \,\forall\, k \in K,\ \implies \mathbf{x} \textit{ is an outlier.}$$

Therefore, $P(\mathbf{x}|k)$ and $P(k|\mathbf{x})$ map the probability functions of belonging to each cluster and the regions formed by outliers.

Figures 4.8 and 4.9 depict the probability maps using the five clusters detected in the QC_{knn} solution (figure 4.7) with a JS=0.850; in the probability map of cluster membership, $P(K|X)$, there is a small cluster (brown colour) that is covered by other clusters with higher probabilities, therefore it is an empty cluster. Figure 4.9 is a top-down projection of the probability map that shows how the cluster membership is defined without using SGD.

Figure 4.10 shows the maximum probability $P(X|K)$ that the model can assign to each region, allowing a tool for outlier detection, $max_K\ P(X|K)$.

Something similar happens with the solutions of QC^{prob}_{cov}, where the algorithm detects seven clusters, but the probabilistic allocation makes a reduction to four clusters.

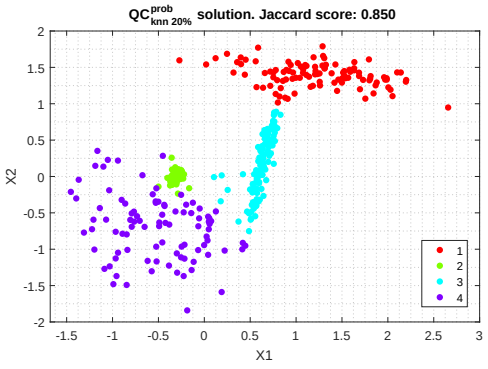


FIG. 4.7 – Solution for $QC^{prob}_{knn\,20\%}$

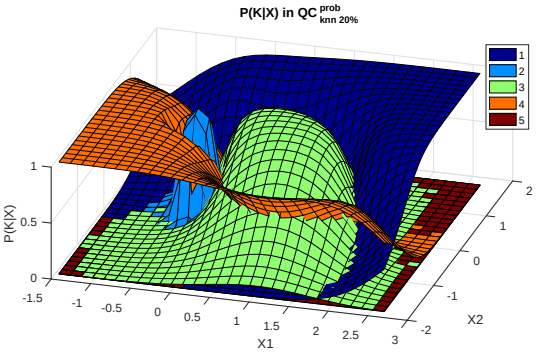


FIG. 4.8 – $P(K|X)$ of $QC^{prob}_{knn\,20\%}$

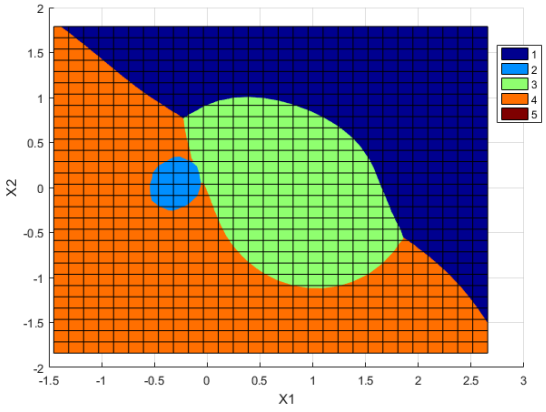


FIG. 4.9 – Projection of $P(K|X)$

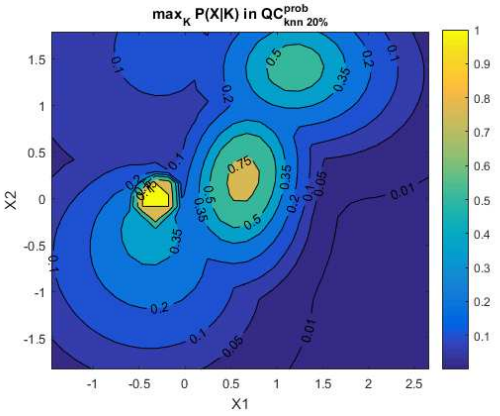


FIG. 4.10 – QC heat map based on $max_K\ P(X|K)$

4.2.5 Performance assessment

4.2.5.1 Average Negative Log-Likelihood, ANLL

This section deals with a major contribution of the chapter, namely, an unsupervised method to select the remaining free hyper-parameter: %KNN. To recall, each observation is allocated to the cluster k with the highest $P(k|\mathbf{x})$, cluster k_w. As a result the observation i is allocated with a probability $P(k_w|\mathbf{x}_i)$.

The quantity $P(k_w|\mathbf{x}_i)$ competes against the probabilities associated with the other clusters in $\mathbf{x}_i$. Rewriting eq. 4.29 in terms of probability marginalization:

$$P(k_w|\mathbf{x}) = \frac{P(k_w, \mathbf{x})}{P(\mathbf{x})} = \frac{P(k_w, \mathbf{x})}{\sum_k P(k, \mathbf{x})} \tag{4.32}$$

Regions close to the cluster boundaries have low values of $P(k_w, \mathbf{x})$, because there are at least two clusters with similar probabilities. Therefore, the best models have the highest value of $P(k_w|\mathbf{x})$ for a high number of observations. This corresponds to the likelihood of cluster membership, given by:

$$LL(K|\mathbf{X}) = log\left(\prod_i^n P(k_w|\mathbf{x}_i)\right) = \sum_i^n log\left(P(k_w|\mathbf{x}_i)\right) \tag{4.33}$$

To normalize the score in the range $[0, 1]$, the average negative log-likelihood (ANLL) may be used:

$$ANLL(K|\mathbf{X}) = \frac{-\sum_i^n log\left(P(k_w|\mathbf{x}_i)\right)}{N} \tag{4.34}$$

The lower the ANLL, the better the model fit. Its value clearly depends on the length scale parameter, %KNN, because the length scale controls the number of clusters and the smoothness of the wave function. A representation of ANLL against %KNN will, in general, have some regions where the ANLL score is minimized, so that $P(k_w|\mathbf{x})$ is maximized, obviously avoiding the trivial solution of a single cluster covering all of the data, which takes the lowest possible value (ANLL = 0). The ANLL surface has minima due to trivial solutions that corresponds to those length scales equal or greater than a threshold that produces a single cluster model, reflecting the gross structure of the data rather than the component clusters. However, these are easily recognised and can be discarded.

The ANLL provides an unsupervised figure of merit which was empirically found to be highly correlated with the supervised JS. Therefore, it can be used as a measure of the clustering performance without the need for prior information about the number of clusters or their composition. Figure 4.11 shows ANLL and JS for different length scales in QC_{cov}^{prob}, to illustrate their correlation. In addition, the ANLL vs %KNN plot reveals the hierarchical structure of the data, where an abrupt change in ANLL means a significant change in the data structure, such as a variation in the most relevant clusters. The bottom plot of figure 4.11 shows how the number of clusters depends on the length scale, although the QC_{cov}^{prob} considerably cushions the fluctuation compared with the original QC.

For the artificial data set #1 and QC_{cov}^{prob}, the Pearson's linear correlation coefficient between Jaccard score and ANLL is $\rho = -0.776$, p-value < 0.001.

4.2.5.2 Extended ANLL score

The extended ANLL score improves ANLL by taking account of the threshold parameter E_{th} that controls the hierarchical solutions defined in each %KNN by setting a threshold to merge two clusters if the maximum potential difference between their centroids along the shortest path is less than E_{th}. A more detailed explanation is presented in Section 4.2.6.

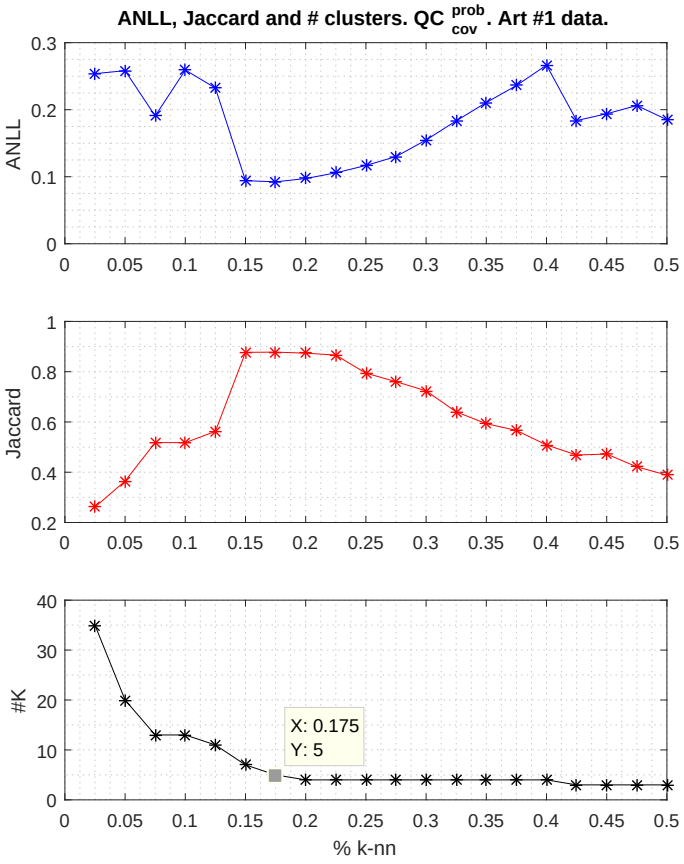


FIG. 4.11 – QC_{cov}^{prob} ANLL, JS and #K respect to %KNN variation in dataset #1

By default, the ANLL score uses a fixed E_{th} that depends on the SGD convergence criteria in the last iteration:

$$E_{th} = max\Big(\epsilon_V, max\big(\Delta V(\mathbf{x}_{iter\,max})\big)\Big) \tag{4.35}$$

This E_{th} corresponds with the lowest possible bound, so that any accidental cluster merging is avoided.

Figure 4.12 shows an enhanced representation of ANLL, including its relationship with E_{th}. To avoid confusion with non-trivial solutions, scores associated with a trivial solution are assigned the highest ANLL score.

The interpretation of the ANLL plots is partly subjective, as the plots give an indication of the clustering structure in the data which may be multi-level when the data are hierarchical. The algorithm 1 describes the steps must be taken to identify interesting PQC solutions, including those that correspond with hierarchical solutions.

Algorithm 1 Procedure of PQC hyper-parameters selection from ANLL plot

Inputs: collection of PQC models fitted to $\mathbf{X}$ and characterized by their hyper-parameters $\%knn$ and E_{th}
ANLL $\leftarrow function\,(\%knn, E_{th})$ ▷ Goodness of fit score for each model
Plot $ANLL$ as a function of $\%knn$ and E_{th}

procedure LocalMinimaKnn(ANLL, $\%knn, E_{th}$)
    $E'_{th} \leftarrow min(E_{th})$
    $\Delta E'_{th} \leftarrow$ Small E_{th} variation
    $Parameters_1 \leftarrow$ Empty list
    for $\%knn' \leftarrow min(\%knn), max(\%knn)$ **do**
        if ANLL $(\%knn', E'_{th})$ is local minima **then**
            if ANLL $(\%knn', E'_{th} + \Delta E'_{th}) \approx$ ANLL $(\%knn', E'_{th})$ **then**
                Model with $(\%knn', E'_{th})$ is an interesting solution
                $Parameters_1 \leftarrow$ Append $(\%knn', E'_{th})$
            end if
        end if
    end for
    return $Parameters_1$
end procedure

procedure LocalMinimaEth(ANLL, $\%knn, E_{th}$)
    $\Delta E'_{th} \leftarrow$ Small E_{th} variation
    $Parameters_2 \leftarrow$ Empty list
    for $\%knn' \leftarrow min(\%knn), max(\%knn)$ **do**
        for $E'_{th} \leftarrow min(E_{th}), max(E_{th})$ **do**
            $\Delta E'_{th} \leftarrow$ Small E'_{th} variation
            $\Delta\%knn' \leftarrow$ Small $\%knn'$ variation
            if ANLL $(\%knn' \pm \Delta\%knn', E'_{th} \pm \Delta E'_{th})$ is $\approx$ absolute minimum **then**
                Model with $(\%knn', E'_{th})$ is an interesting hierarchical solution
                $Parameters_2 \leftarrow$ Append $(\%knn', E'_{th})$
            end if
        end for
    end for
    return $Parameters_2$
end procedure

$SelectedParameters \leftarrow Parameters_1 + Parameters_2$ ▷ Interesting solutions to check

Figure 4.12 shows extended ANLL score versus E_{th} and %KNN. For low E_{th} values, being $E_{th} = 0.001$ the default value, the figure 4.12 presents the same pattern of ANLL observed in figure 4.11. This pattern is kept up to some E_{th}, at which point the clusters start to merge. In the range of values where the pattern is constant the proper %KNN for the QC model can be identified. Then, looking for stable regions of low ANLL values for high E_{th}, where hierarchical solutions could also be good solutions, these would have fewer clusters; this phenomenon does not appear in artificial data set #1, but it will be shown for data set #2 in Section 4.4.

The ANLL scores always diminish when E_{th} increases, because it is an implicit reduction of the number of clusters. That explains why the ANLL score is less reliable when there are few clusters, because the trivial solution, with a unique cluster, always leads to ANLL equal to zero. To avoid confusion with non-trivial solutions, scores associated with a trivial solution are assigned with the highest ANLL score (we call this ANLL modification as ANLLmod).

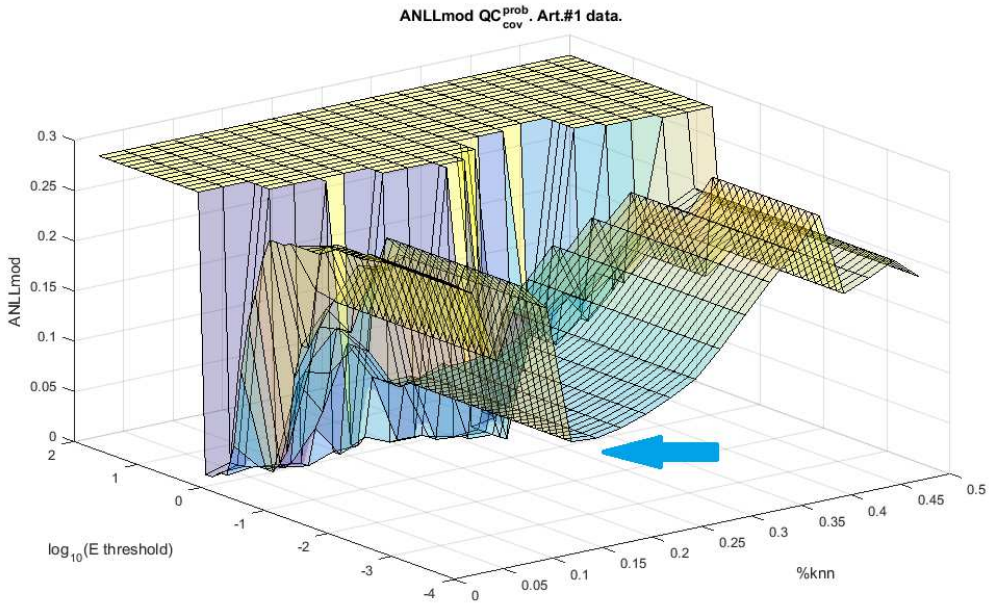


FIG. 4.12 – QC_{cov}^{prob} ANLLmod respect to %KNN and E_{th} variation in dataset #1

4.2.6 Improved Cluster Allocation

Once the gradient descent converges, the observations are allocated to particular potential wells. The first step is to identify the groups of closer observations that lie in a potential well as a cluster. One of the most robust methods is to apply a community detection algorithm, such as Modularity Maximization [117], since the observations are densely located in each potential well the algorithm will identify the communities easily.

To apply the community detection algorithm, the data is transformed into a network using pairwise distances, where the Euclidean distances include the potential values as an extra feature of the data. The adjacency matrix is based on a similarity matrix with Gaussian radial kernel of the mentioned distances.

In order to implement a community detection, an auxiliary data formed by scaling the original data is created, in addition to the potential values scaled as if they were an extra dimension:

$$\mathbf{X}_{aux}^{d+1} = \left[\frac{\mathbf{x}_i}{\lambda_{\mathbf{x}}}, \frac{V(\mathbf{x}_i)}{\lambda_V}\right] = \mathbf{Y} \tag{4.36}$$

where each scaling factor has the purpose to make similar the magnitudes of both terms:

$$\lambda_{\mathbf{x}} = \frac{\sum_{i=1}^{n} |\mathbf{x}_i|}{n} \qquad \lambda_V = \sqrt{\frac{\sum_{i=1}^{n} V(\mathbf{x}_i)^2}{n}} \tag{4.37}$$

With the new data $\mathbf{Y}$ of dimensionality $d+1$, each observation contains the information of its space position and its potential equally weighted. To build the adjacency matrix of the network, a pairwise distance is computed to build a similarity matrix using a Gaussian radial kernel:

$$A_{i,j} = exp\left(-\frac{dist\left(\mathbf{y}_i, \mathbf{y}_j\right)^2}{\sigma_G^2}\right) \tag{4.38}$$

where σ_G^2 is a network locality parameter, empirically adjusted to $\sigma_G = 0.1$ in order to be large enough to capture all the clusters around the space of 0.1 radius. The Modularity Maximization community detection algorithm was implemented by means of the Community Detection Matlab Toolbox by Athanasios Kehagias[2].

Once the communities have been detected, each community being a cluster, the centroids of each cluster are computed by simply averaging their positions.

Now, with the list of clusters and centroids, the goal is to evaluate which is the minimum difference of potential required to cross from *centroid-i* to *centroid-j*. Let us call "energy" the minimum difference of potential. The purpose of this step is to merge some nearby

[2]https://es.mathworks.com/matlabcentral/fileexchange/45867-community-detection-toolbox

sub-clusters that actually belong to the same cluster but were split by the community detection algorithm.

The network is extended with the list of centroids creating a fully connected network, where each node represents an observation or a centroid, and each edge represents the distance between the nodes based on the potential values, $V(\mathbf{x})$.

Using the Dijkstra algorithm [55] to find the shortest-path in the network, a pairwise measure of the potential differences between centroids (nodes) can be built. For instance, in order to go from the i-th node to the j-node, the shortest path found will require at least an "energy" in terms of the potential units, as:

$$\Delta V(\text{path})\Big|_{\text{node}_i}^{\text{node}_j} = max\left(V_{\text{path}}\right) - V(\text{node}_i) \tag{4.39}$$

It should be remarked that the opposite, going from the j-th node to the i-th node may have a different ΔV.

With this information,a pairwise network between nodes with the minimum energy to go from one to another node is built. The next step is to establish a threshold energy to merge any cluster with very similar potential. This procedure is performed to avoid certain situations where several potential wells are connected, for instance by a valley, and hence they have $\Delta V \approx 0$ but not strictly 0. Therefore, one could consider as the same cluster any pair of potential wells that satisfy:

$$\text{If}\,\Delta V\Big|_{\text{node}_j}^{\text{node}_i} \leq E_{th} \implies merge\ (\text{cluster}_j, \text{cluster}_i) \tag{4.40}$$

The E_{th} parameter controls the minimum potential difference allowed between two potential wells along their shortest path, $\Delta V\Big|_{\text{node}_j}^{\text{node}_i}$; they are considered as different clusters if $\Delta V\Big|_{\text{node}_j}^{\text{node}_i} > E_{th}$.

On the other hand, if E_{th} is progressively increased, more clusters will be merged, up to some point where all the clusters are merged. Thus, this parameter allows control of the hierarchical structure of the clusters for a specific QC solution defined by %KNN, in other words, keeping the same wave function and same potential shape, the clusters can be hierarchically merged as a function of potential well differences.

Is there a E_{th} lower bound? Yes, it is related to the SGD convergence criteria, that was $max\big(\epsilon_V, max\big(\Delta V(\mathbf{x}_{iter\,max})\big)\big)$, where ϵ_V was set to 0.001 in normalized data, and

$max\big(\Delta V(\mathbf{x}_{iter\,max})\big)$ is the maximum ΔV reached in the SGD if it has been stopped by number of iterations instead of by reaching the convergence criteria, ϵ_V. Then:

$$min\Big(\Delta V\big|_{node_j}^{node_i}\Big) \geq max\Big(\epsilon_V, max\big(\Delta V(\mathbf{x}_{iter\,max})\big)\Big) \tag{4.41}$$

On the other hand, is there a E_{th} upper bound? The theoretical answer is no, but if the data is normalized it is very unlikely to obtain potential differences greater than 100 in energy units. Most of the time, all the clusters are merged when $E_{th} \approx 10$.

Knowing the upper and lower bounds, $10^{-3} \leq E_{th} < 10^2$, the ANLL score can be mapped with respect to the E_{th} in addition to the %KNN variable, viewing a whole picture of all the QC solutions. In terms of computational time, checking the solutions of different E_{th} is not expensive compared to computing different %KNN. By way of example, figure 4.12 shows the ANLL score respect to the E_{th} and %KNN.

Analysing the ANLL behaviour with respect to E_{th}, one may observe that ANLL always decreases as E_{th} increases because more clusters are merged. When E_{th} is high enough to merge all clusters, the ANLL score is equal to zero. To avoid confusions between the low ANLL values (good solutions) and the trivial solutions (a single cluster), the ANLL values of the trivial solutions have been masked with an arbitrary offset greater than zero, for instance, with the maximum ANLL value in the plot. Doing that the low values corresponding to interesting solutions from the trivial solutions are easily identified, as one may observe in figure 4.12.

To interpret the ANLL plots and extract solutions of interest, the instructions detailed in Section4.2.5.2 must be followed. In the case of finding a stable region with high E_{th} and a specific %KNN, the distribution of potential differences between potential wells can be checked; let say $\Delta E \equiv \Delta V(path)\Big|_{node_j}^{node_i} \quad \forall \quad i, j$. The histogram of ΔE can highlight a hierarchical structure if there is a multi-modal distribution with regions completely separated. Also, if the E_{th} needed to merge each cluster is represented the big energy jumps can be observed, where two clusters with different structures are merged.

To illustrate that case, the ΔE is compared between the data sets #1 and #2 in figure 4.13. Left figures show the E_{th} associated with each solution for a specific number of clusters. Right figures show the histogram of the pairwise differences between potential wells (ΔE) for a specific solution, %KNN = 10%. Top figures correspond with the artificial data set #1 and bottom figures to the artificial data set #2 (spirals). The data set #1 (top-left plot) shows how the E_{th} needed to merge clusters increases progressively without big jumps, and in the histogram (top right) there is not a clear multi-modal

distribution completely separated, this is an indication that there is not an underlying hierarchical structure, and no solution should be considered with higher E_{th} in the ANLL plot (figure 4.12). In the spiral case, there is a clear behaviour of the existence of an underlying structure. The bottom-left plot shows a big jump in the E_{th} to merge the last two clusters into only one cluster. The ΔE distribution (bottom right) clearly shows two separated groups of modes, the left group corresponds with ΔE between intra-spiral clusters, and the right group corresponds with ΔE between inter-spiral clusters. In this case, solutions with high E_{th} should be considered in the ANLL plot (figure 4.22).

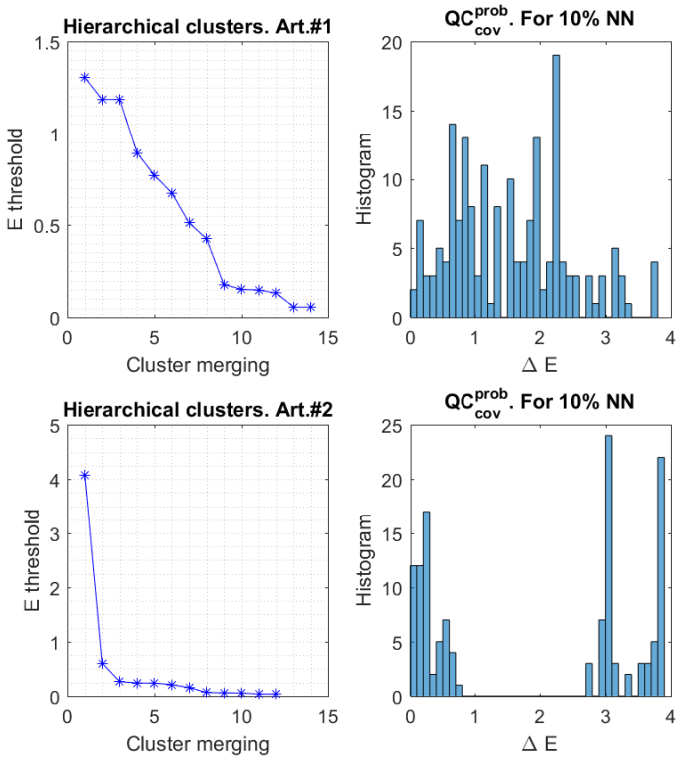


FIG. 4.13 – QC E_{th} distribution between cluster centroids for data #1 and #2

4.2.7 Selection of local-covariance threshold

The most appropriate local-covariance threshold for the QC_{cov}^{prob} model is obtained by mapping ANLL (Section 4.2.5), JS and the number of clusters found (K), as a function of the %KNN and the threshold ratio r. Finally, the neighbours considered in local-covariance threshold are given by:

$$\%K'NN = r\ (\%KNN)$$

The tests were carried out for the artificial data set #1. The ANLL map is presented in figure 4.14, from which the following conclusions can be drawn:

- The valley of lowest (best) ANLL values lie around 20%KNN. However, the most clear valley corresponds with values that range from 0.5 to 1.0; lower ratios lead to a noisy response of ANLL.
- If %KNN $>$ 40%, only three clusters are found, and it corresponds, in turn, with a too biased model, which remains practically invariant to the value of the threshold ratio.
- The best ANLL values are for ratios from 0.9 to 1.0.

Figure 4.15 shows the JS map, and also provides useful information:

- JS is less affected by the threshold ratio than ANLL. It is, though, more sensitive to the %KNN.
- The best solutions are located around 20%KNN, as in the case of ANLL.
- There is a common drop in performance when the ratio decreases; the best values for the highest ratios are $r \in [0.9, 1]$

Figure 4.16 shows the number-of-clusters map. The main conclusions are, as follows:

- Both variables, %KNN and threshold ratio are inversely correlated with the number of clusters. This effect was already mentioned in Section 4.2.3; low values of %KNN or %K'NN involve a reduced interaction of each kernel, thus creating an excess of local minima and sub-clusters. The interesting point is to observe that the threshold ratio has a similar influence to %KNN for creating sub-clusters.
- As the best solutions are for high values of the threshold ratio, one can conclude that an excess of sub-clusters decrease the performance.

In summary, in order to obtain the most simple solution, avoiding spurious sub-clusters, the best threshold ratio is $r \in [0.9, 1.0]$; for the sake of simplicity, $r = 1.0$ is probably the most sensible choice. That means %K'NN = %KNN.

4.3 Data description

Two challenging artificial data sets and three real-world data sets were employed to test the theoretical hypotheses and evaluate the clustering performance for the methods reviewed in this chapter.

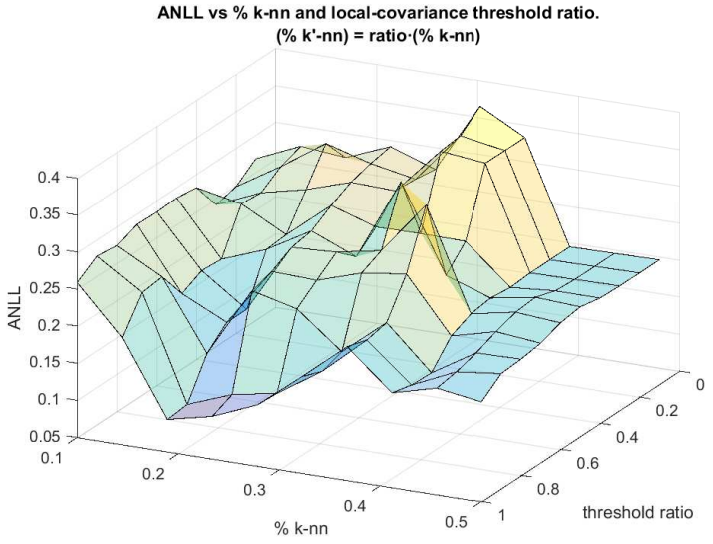


FIG. 4.14 – QC_{cov}^{prob} ANLL vs %KNN and E_{th} for dataset #1

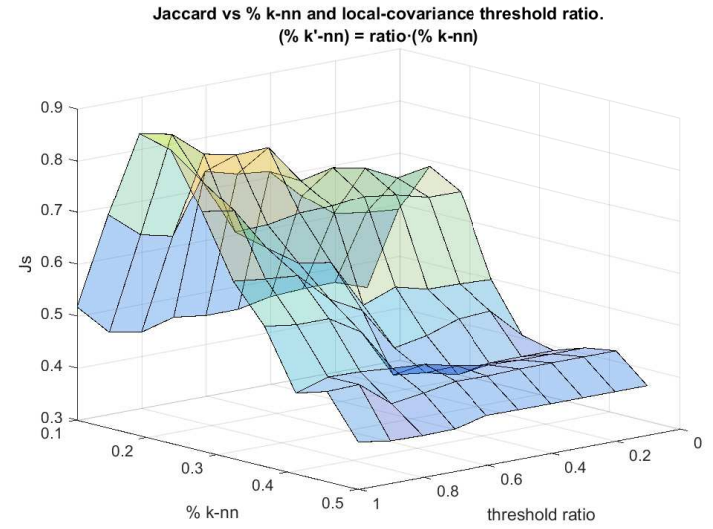


FIG. 4.15 – QC_{cov}^{prob} JS vs %KNN and E_{th} for dataset #1

4.3.1 Data set #1 (artificial): Local densities

This data set has two main characteristics which challenge clustering algorithms: first, there are two clusters with cigar shapes; second, there are two clusters partially overlapped but with different local densities. The original QC was able to detect anisotropic clusters, but it is less able to discriminate clusters with different local densities. The data set is two-dimensional to aid visualization and comprises four clusters with 100 observations each.

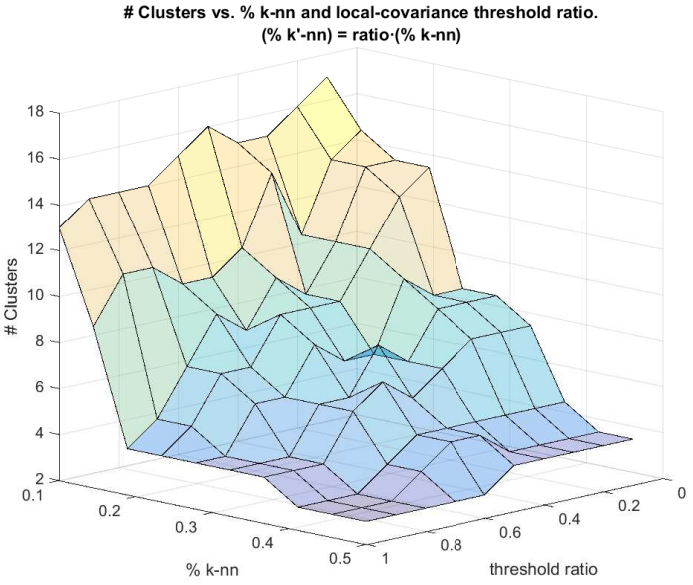


FIG. 4.16 – QC_{cov}^{prob} cluster number vs %KNN and E_{th} for dataset #1

4.3.2 Data set #2 (artificial): Two spirals

This is a spiral data set with standard deviation in the first spiral of 0.1 and 0.025 in the second spiral. The data set is also two-dimensional. Each cluster has 200 observations, 400 in total.

4.3.3 Data set #3 (real): Crabs

This well-known data set was used in the original QC paper, [118]. The Crabs' data set describes five morphological measurements on 50 crabs of each of two colour forms and both sexes, of the species Leptograpsus variegatus collected at Fremantle, W. Australia. In total there are 200 observations and four different labels, two for gender and two for each species. To compare the results with the original paper, PCA has been applied, selecting only the two first principal components (PCs) shown in figure 4.17.

4.3.4 Data set #4 (real): Olive oil

The *Italian olive oil* data set [119] consists of 572 observations and 10 variables (figure 4.18). Eight variables describe the percentage composition of fatty acids found in the lipid fraction of these oils, which is used to determine their authenticity. The remaining two variables contain information about the classes, which are of two kinds: three "super-classes" at country level: North, South, and the island of Sardinia; *and* nine collection area classes: three from the Northern region (Umbria, East and West

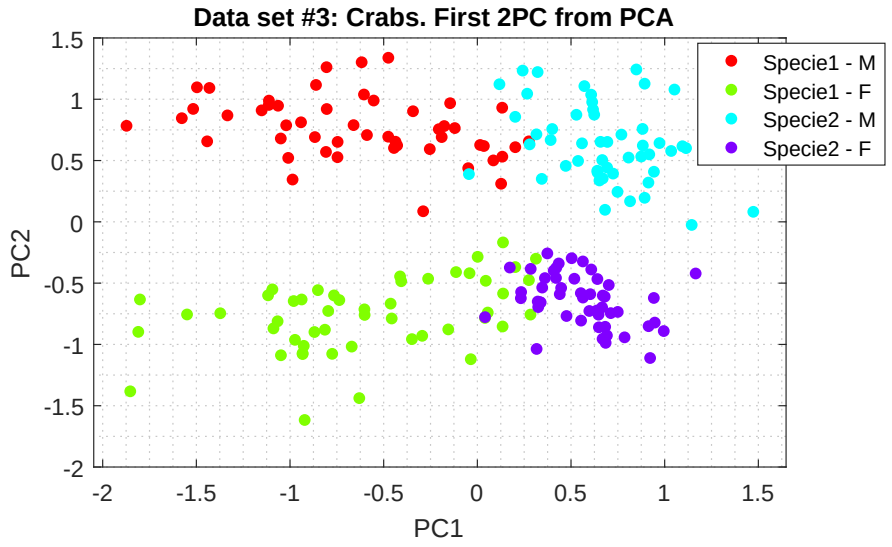


FIG. 4.17 – Crabs data set by principal components

Liguria), four from the South (North and South Apulia, Calabria, and Sicily), and two from the island of Sardinia (inland and coastal Sardinia). This data set is characterised by a hierarchical structure with three large regions or nine sub-areas, which makes it especially appealing for testing clustering algorithms.

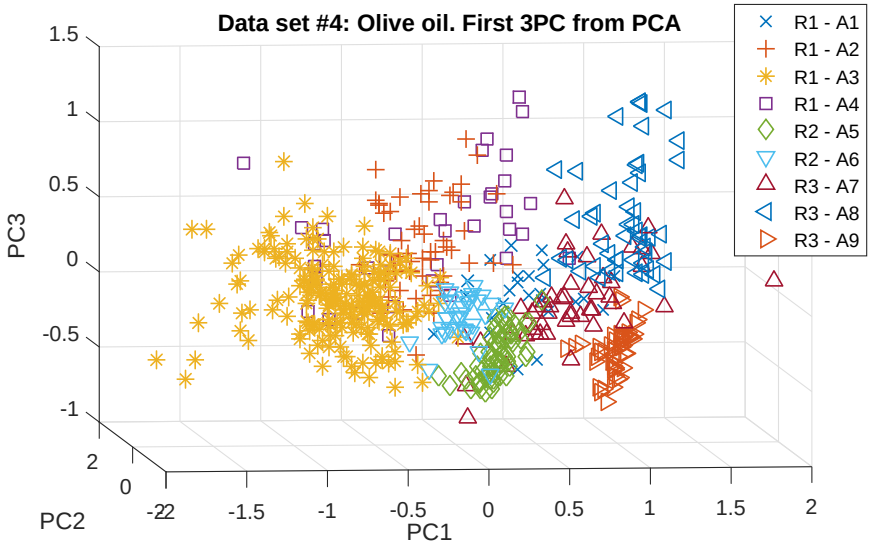


FIG. 4.18 – Olive oil data set

4.4 Results

This section evaluates the extent to which the ANLL score can determine the most suitable %KNN to maximize the JS, highlighting the peculiarities of each data set and comparing the results of both models, QC_{knn}^{prob} and QC_{cov}^{prob}. As ANLL is biased, favouring

fewer clusters, when several local minima appear in ANLL, the ones associated with lower %KNN values should have priority over the ones with higher %KNN values, because ANLL tends to be smaller as the number of clusters decreases.

The results tables list the following information:

- Column 1: data set number and identifies the QC model.
- Column 2: score employed to select the quantile (%KNN), firstly the supervised choice according to the best Jaccard score, then the unsupervised option based on the local minima found in ANLL, starting from the lowest %KNN values in case there were more than a local minimum, and finally checking if the extended ANLL has an stable region increasing the E_{th} parameter.
- Column 3: the E_{th} parameter, by default is used $E_{th} = 0.001$, but then the extended ANLL plot is analysed to find stable ANLL regions with solutions of higher hierarchical order.
- Column 4: length scale parameter in quantiles (%KNN)
- Column 5: number of clusters (#K)
- Column 6: ANLL score
- Column 7: Jaccard score - for the Olive oil data there are two possible classifications, with 3 regions or 9 subregions of Italy.
- Column 8: Cramerś V score - for the Olive oil data there are two possible classifications, as above.
- Column 9: Pearson's linear correlation coefficient between ANLL score with $E_{th} = 0.001$ and the Jaccard score.
- Column 10: The p-values of correlation coefficient for testing the hypothesis of no correlation against the alternative that there is a non-zero correlation.

4.4.1 Data set #1: Local densities

Table 4.1 shows that both models, QC^{prob}_{knn} and QC^{prob}_{cov}, perform similarly for this data set. QC^{prob}_{knn} has the correct number of clusters, four, with a $JS = 0.85$, however QC^{prob}_{cov} with five clusters has a slightly better value, $JS = 0.88$. In both cases, the ANLL corresponds with the Jaccard score. On the other hand, there is not a stable region of low ANLL with high E_{th} values, therefore no hierarchical solution was considered.

TABLE 4.1 – Results data set #1: Local densities

Data #1	*Score*	E_{th}	*%KNN*	*#K*	*ANLL*	*JS*	C_v	$\rho_{E_{th}}$	*p-val*
QC_{knn}^{prob}	Best JS	0.001	17.5	4	0.082	0.85	0.94	-0.81	1.5E-5
	Best ANLL	0.001	17.5	4	0.082	0.85	0.94	-	-
	ANLL stable high E_{th}	No	-	-	-	-	-	-	-
QC_{cov}^{prob}	Best JS	0.001	17.5	5	0.092	0.88	0.96	-0.87	6.0E-7
	Best ANLL	0.001	17.5	5	0.092	0.88	0.96	-	-
	ANLL stable high E_{th}	No	-	-	-	-	-	-	-

4.4.2 Data set #2: Two spirals

The results achieved for this data set are of particular interest. The spirals are not mixed although they are broken-up into sub-clusters due to the value of E_{th}, being too small (a further discussion is provided in the Section 4.2.6).

Figure 4.19 shows ANLL, Jaccard score, C_v and number of clusters obtained by QC_{knn}^{prob} for data set #2. ANLL splits the graph into two regions separated by a value of KNN equal to 22.5%; at the left side, the spirals are not mixed but broken up; while at the right side the spirals are mixed but there are only two clusters. Obviously, an external supervision would prefer not-mixed spirals. Actually, JS is quite low because it is not a good metric for this data set as it does not attribute any importance to the fact that the spirals are not mixed, it only measures similarity with the true labels. To address this issue, the Cramer's V-index (C_v) was used, which is a normalized version of the standard chi-square test for contingency tables; C_v measures the concordance between different cluster allocations for a given number of clusters, detecting when the spirals are mixed if $C_v < 1$.

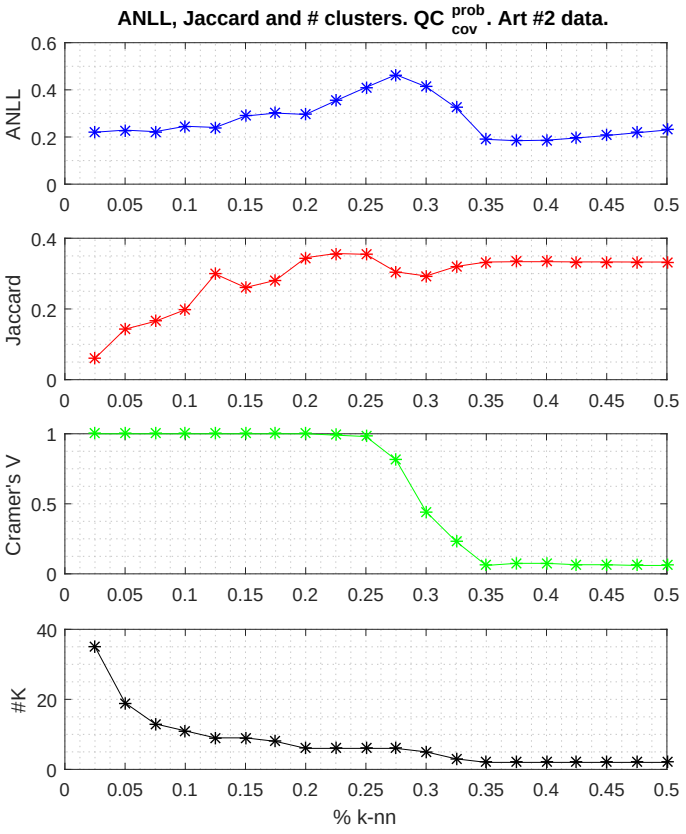


FIG. 4.19 – QC_{knn}^{prob} ANLL, JS, C_v and cluster number for data set #2

C_v shows that the spirals are not mixed until 25% KNN for QC_{cov}^{prob}, however the spirals are fragmented into sub-clusters. Length scales greater than 25% KNN make the potential too smooth and the potential wells mix the spirals.

If guided only by the ANLL score in figure 4.19, two local minima would be selected, the first at $7.5\%KNN$ and the second at $35\%KNN$, keeping E_{th} with the default value (0.001). Both solutions are illustrated in figures 4.20 and 4.21. The QC_{cov}^{prob} solution of the left figure 4.20 produces spirals that are not mixed but each one is fragmented in sub-clusters. The QC_{cov}^{prob} solution of the right figure 4.21 has a the length scale too big to preserve the spirals not mixed, producing the right number of clusters, two, but the spirals are mixed. These cases show the need of the extended ANLL plots.

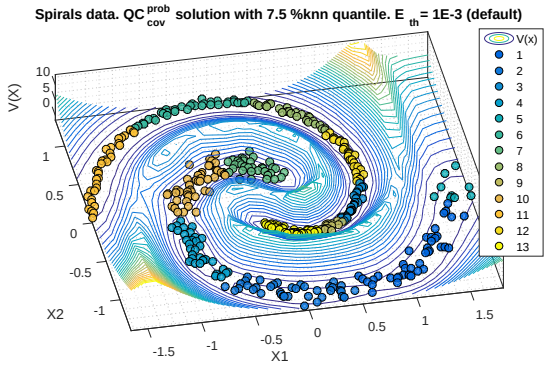


FIG. 4.20 – QC spirals solution for $QC_{knn\,20\%}^{prob}$

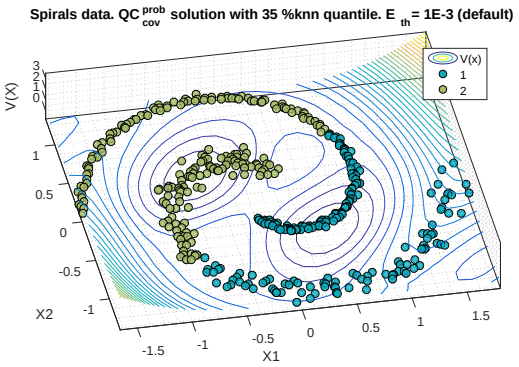


FIG. 4.21 – QC spirals $P(K|X)$

In order to find the best solution ($JS = 1$), where the spirals are neither mixed nor fragmented, the value of E_{th} should be increased until reaching the region of low ANLL values, as shown in figure 4.22. For this case, the extended ANLL score shows a stability region for high E_{th} values. This region offers a solution based on low length scales where the sub-clusters are merged hierarchically to form the two spirals without being mixed. The ANLLmod plot indicates three regions of interest: local minima with small length scale (blue arrow), local minima with higher length making a too smooth potential (green arrow), and the stable region of high E_{th} offering the most interesting solution (red arrow). The best solution depicted in figure 4.23, is achieved in regions with low values of %KNN (< 0.20) and higher E_{th} ($\in [10^{-1}, 10^{0}]$).

Although ANLL is not highly correlated with JS along the E_{th} axis direction, a stable region of low ANLL with high E_{th} implies an underlying hierarchical structure that produces a good JS. This effect generalizes well when the underlying structure is very distinct, as in the case of two spirals separated by high potential walls in the potential space. Additionally, it is relevant to analyse the distribution of ΔE for detecting separate distributions. - see figure 4.13 at the end of Section 4.2.6 for further details.

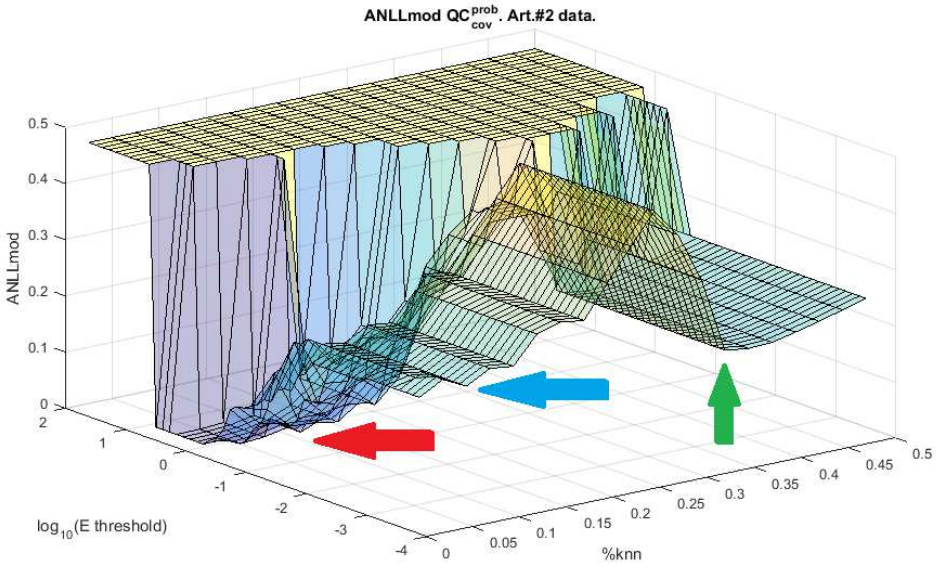


FIG. 4.22 – ANLL solutions for QC spirals data

Based on the parameters identified by the extended ANLL plot in the stable region in figure 4.22, it is apparent that the perfect solution for the spiral case is obtained by increasing the E_{th} with low %KNN. Since the JS is not ideally suited for this data set, the expected inverse correlation with ANLL is not present for both, QC_{cov}^{prob} and QC_{knn}^{prob}, in Table 4.2.

Table 4.2 shows the results obtained with both models. The stability region varies depending on the QC model, but can be inspected visually using the ANLL plot.

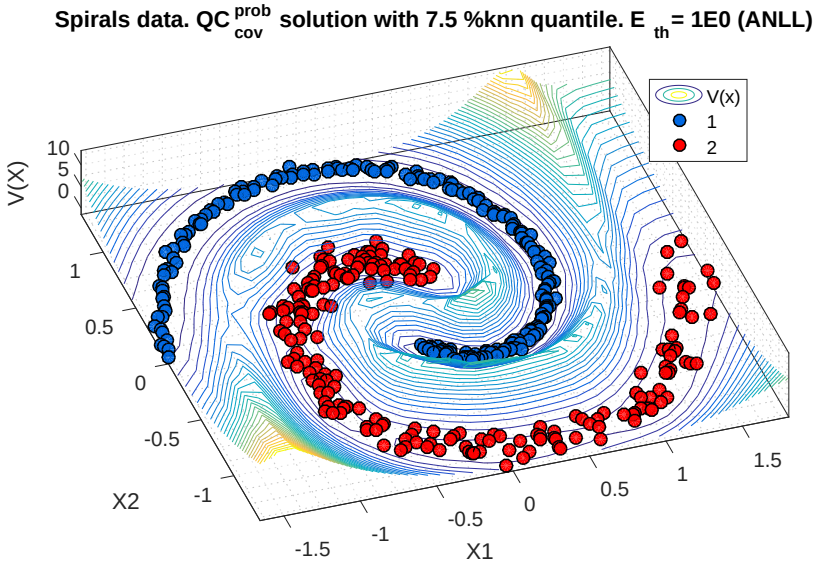


FIG. 4.23 – Spirals solution based on the ANLL stable region parameters

TABLE 4.2 – Results table of data set #2: Two spirals

Data #2 Spirals	*Score*	E_{th}	*%KNN*	*#K*	*ANLL*	*JS*	C_v	$\rho_{E_{th}}$	*p-val*
QC^{prob}_{knn}	Best JS	0.001	47.5	1	0.510	0.50	-	0.60	0.005
	Best ANLL1	0.001	7.5	14	0.237	0.16	1.00	-	-
	Best ANLL2	0.001	35.0	2	0.229	0.33	0.06	-	-
	ANLL stable at high Eth	$[0.2, 0.8]$	$[2.5, 10]$	2	6.8E-5	**1.00**	1.00	-	-
QC^{prob}_{cov}	Best JS	0.001	22.5	6	0.354	0.36	0.99	0.19	0.412
	Best ANLL1	0.001	7.5	13	0.223	0.17	1.00	-	-
	Best ANLL2	0.001	35.0	2	0.190	0.33	0.06	-	-
	ANLL stable at high Eth	$[0.5, 1.5]$	$[2.5, 20]$	2	1.0E-5	**1.00**	1.00	-	-

TABLE 4.3 – Results table of data set #3: Crabs

Data #3 Crabs	*Score*	E_{th}	*%KNN*	*#K*	*ANLL*	*JS*	C_v	$\rho_{E_{th}}$	*p-val*
QC^{prob}_{knn}	Best JS	0.001	17.5	4	0.110	0.74	0.90	-0.83	5.7E-6
	Best ANLL	0.001	17.5	4	0.110	0.74	0.90	-	-
	ANLL stable at high Eth	No	-	-	-	-	-	-	-
QC^{prob}_{cov}	Best JS	0.001	15.0	4	0.126	0.70	0.89	-0.88	2.8E-7
	Best ANLL	0.001	15.0	4	0.126	0.70	0.89		
	ANLL stable at high Eth	No	-	-	-	-	-	-	-

4.4.3 Data set #3: Crabs

For the Crabs' data set, ANLL also obtains the appropriate %KNN corresponding with the best JS. Table 4.3 shows that QC_{knn}^{prob} leads to $Js = 0.70$ and QC_{cov}^{prob} to $Js = 0.74$, respectively. In relation to the extended ANLL score there are no stable hierarchical solutions.

4.4.4 Data set #4: Olive oil

The olive oil data set is especially challenging due to its double structure and the fact that some clusters are partially overlapped. Table4.4 shows the main results for both classifications. JS in bold refer to the value that should be compared to the corresponding ANNL, depending on whether the model is a solution of the 3-class or 9-class problem.

TABLE 4.4 – PQC table results of data set #4: Olive oil

Data #4 Olive	*Score*	E_{th}	*%KNN*	*#K*	*ANLL*	*JS1 JS2*	C_V1 C_V2	$\rho_{E_{th}}$	*p-val*
QC^{prob}_{knn}	Best JS1 3 regions	0.001	12.5	5	0.241	0.77 —	0.83 —	0.08	7.5E-1
	Best JS2 9 regions	0.001	2.5	9	0.167	— 0.73	— 0.98	-0.33	1.5E-1
	Best ANLL1	0.001	7.5	5	0.162	0.64 **0.55**	0.85 0.89	-	-
	Best ANLL2	0.001	22.5	2	0.230	**0.74** 0.36	0.97 0.95	-	-
	ANLL stable at high Eth	No	-	-	-	-	-	-	-
QC^{prob}_{cov}	Best JS 3 regions	0.001	47.5	4	0.231	0.79 —	0.76 —	-0.67	1.4E-3
	Best JS 9 regions	0.001	20.0	8	0.187	— 0.73	— 0.76	-0.52	1.9E-2
	Best ANLL1	0.001	15.0	9	0.175	0.52 **0.72**	0.99 0.72	-	-
	Best ANLL2	0.001	45.0	4	0.220	**0.78** 0.41	0.76 0.81	-	-
	ANLL stable at high Eth	No	-	-	-	-	-	-	-

For the QC^{prob}_{knn}, the first ANLL local minimum is closer to the real 9 regions classification but ANLL does not identify the best length scale available: 7.5%KNN (JS 0.55) instead of 2.5%KNN (JS 0.73). The second ANLL local minimum obtains a similar JS to the best JS possible, although the length scale is quite different: 22.5%KNN instead of 12.5%KNN. Despite not matching exactly with the highest JS, the information provided by the two minima is of paramount relevance, as they point out the two underlying structures, namely, three and nine clusters. The ANLL-JS correlation is quite poor, partly due to ANLL reflecting two behaviours but it is compared with two different JS curves.

However for this dataset, the QC^{prob}_{cov} clearly outperforms QC^{prob}_{knn}, ANLL finds solutions with JS practically as good as the best JS ones, the ANLL-JS correlation is better, and the cluster number is close to the real ones (#K: 4 and 9).

A further detailed explanation can be obtained observing figure 4.24: The algorithm starts with many sub-clusters with the first KNN; it is important to take into account that dealing with more than 100 clusters is computationally very expensive during the cluster allocation because it has to check $100 \cdot 99 = 9900$ possible paths between potential wells (centroids). Then, the number of clusters decreases drastically until obtaining nine clusters in 15% KNN, and it is here where the first local minimum appears in ANLL, matching with the highest Jaccard score for the structure of nine areas. Then, a subtle local minimum appears at 45% KNN, very close to the highest Jaccard score for the structure of three regions of Italy. Lastly, there is another ANLL minimum at 50% KNN; it is not a real solution but an effect of dealing with very few clusters. The best Jaccard for three regions is $JS = 0.73$, and for nine areas is $JS = 0.79$.

In relation to the extended ANLL score, figure 4.25 shows the analysis of the variation of E_{th} for merging clusters. ANLL is represented versus the logarithm of E_{th} and %KNN. There is not any stable region with high E_{th} and low ANLL values and therefore no stable hierarchical solutions. The red arrow points out the 9 regions structure and the green one points out the 3 regions classification.

4.5 Conclusion

This chapter presents a novel framework for detecting the underlying structure in data, within the paradigm of Quantum Clustering (QC). In particular, a merit function to measure goodness of fit is presented in the form of the Average Negative Log-Likelihood (ANLL). This utilises a Bayesian framework to enable optimisation of a control parameter for the estimation of local length scales using set percentages of nearest neighbours.

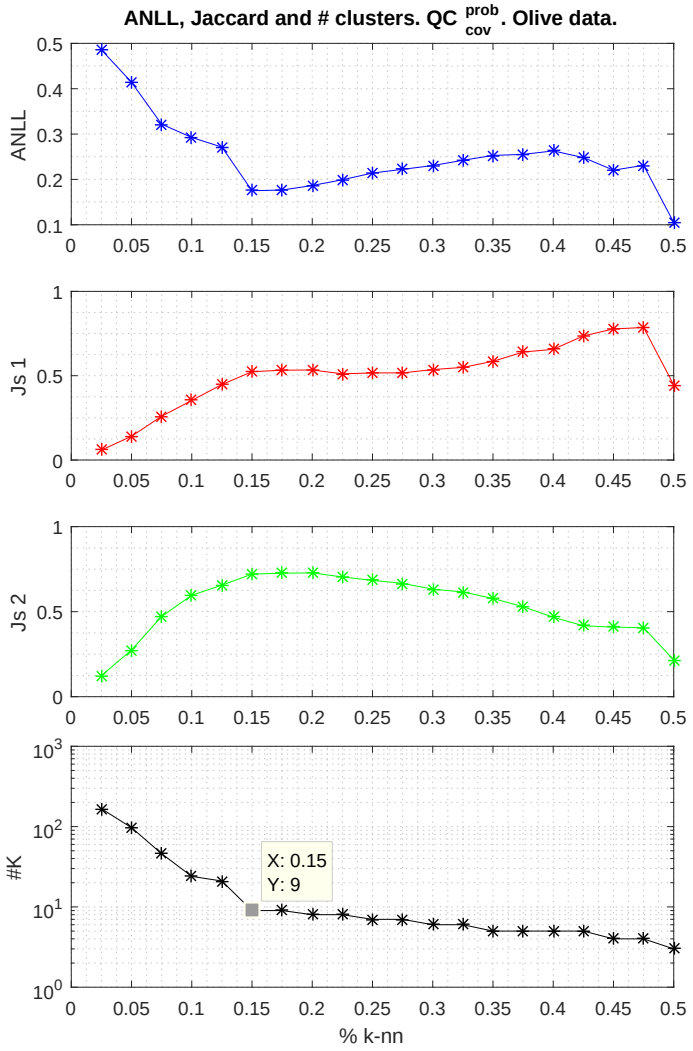


FIG. 4.24 – QC_{cov}^{prob} ANLL, JS, C_v and cluster number for the olive oil data

Local minima of the ANLL are shown empirically to be correlated with the highest values of the Jaccard Score (JS) measured against cluster labels derived from generating density functions, in the case of synthetic data, or known a priori for real-world data set. Therefore ANLL is a useful objective performance index for unsupervised learning. Furthermore, the ANLL provides useful guidance and insight into QC solutions to detect hierarchical structures in the data.

Two new models for Probabilistic Quantum Clustering (PQC) with different levels of computational complexity are proposed. Attending to its simplicity and versatility QC_{knn}^{prob} may outperform QC_{cov}^{prob} in general. However, QC_{cov}^{prob} performs better with more challenging data.

The main limitation of QC_{cov}^{prob} stems from its less smooth potential functions as local-covariance kernels have less superposition effect than spherical kernels. As a consequence

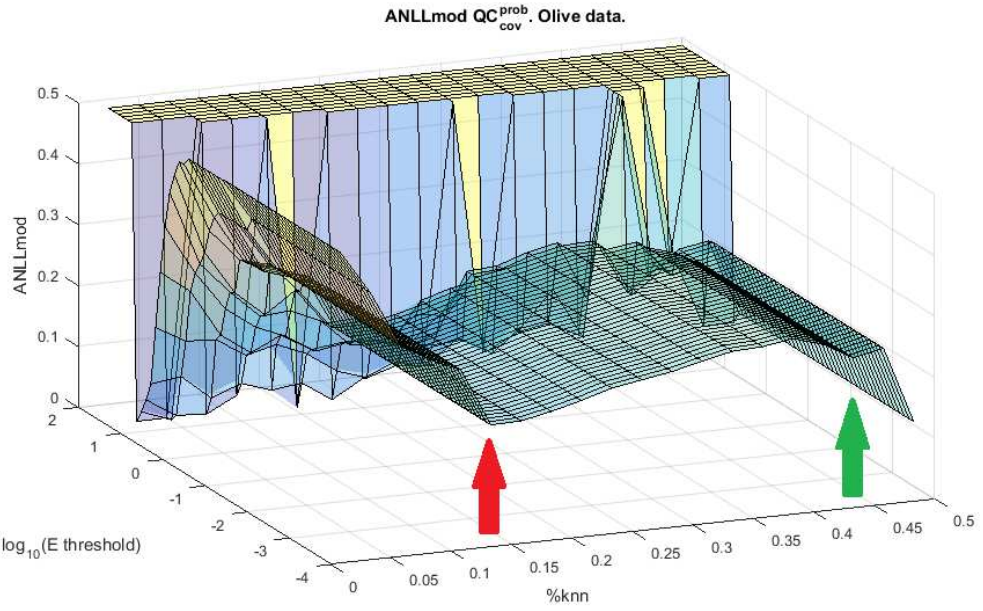


FIG. 4.25 – QC^{prob}_{cov} extended ANLL for Olive oil data

of this:

- QC^{prob}_{cov} needs more iterations in the SGD to achieve the same convergence than QC^{prob}_{knn}, mainly because the potential shape is less convex and the SGD is less effective.
- QC^{prob}_{cov} tends to create more sub-clusters due to the presence of more local minima. This is not an inconvenience in itself because these sub-clusters can fit better the data and then can be later merged in the cluster allocation process. However, if too many sub-clusters appear, the computation time needed to check all the possible paths between all the centroids may be excessive.

The underlying probabilistic framework for QC enables outlier detection as well as the delineation of Bayesian optimal cluster boundaries. QC methods are well-known to have poor performance for high-dimensional data. The proposed framework shares these inherent limitations the root of which lies in the ultra-metric nature of Euclidean distances in high dimensions as well sparsity which causes difficulties for local covariance estimation. This remains an area of further work. Also the algorithm runtime could be improved implementing some parallel computing tool, like GPGPU-based.

In the next chapter, the PQC will be used to cluster the embedded Fisher manifold obtained from a music dataset based on spectral features, where the manifold presents local density variations.

Chapter 5

Conditional Independence Maps

This chapter focuses on data structure learning algorithms that find relationships between features. This chapter has two parts, the first part focuses on developing a systematic methodology for building associative maps based on conditional independencies between features, and from these conditional independence maps (CI-maps) Bayesian networks can be derived. The second part of the chapter exploits these CI-maps for extracting a hierarchical order of variable associations with respect to a target variable. The method is based on bootstrapping several maps in order to identify the most representative CI-map based on bootstrapped data. The CI-map methodology is validated with benchmark datasets, and then applied to a brain tumour data, where the CI-map displays the association of brain tumour types with different metabolites observed with magnetic resonance spectroscopy (MRS).

The schema of the chapter is as follows: this introduction 5.1 makes a brief overview, the chapter continues with the background section 5.2, where there is a more detailed description of the main parameters and policies used in the PC-algorithm. Section 5.3 introduces the benchmark data where the true structure is known, and therefore is used for evaluating the parameters, and also introduces the brain tumours data where the procedure is applied. Then the methodology section 5.4 presents the procedures for assessing these parameters with benchmark data, and then in Subsection 5.4.3 the methodology for selecting the CI-map most representative of the bootstrapped data is described, giving a robust representation of the variables more often associated with a target variable. After this, the results section 5.5 and discussion 5.6 for both parts are presented. The chapter is then completed with final conclusions 5.7.

5.1 Introduction

The starting point of the methodology developed in this chapter is the PC algorithm. In the field of structure finding, the PC algorithm is a well-known constraint-based algorithm used to build a Directed Acyclic Graph (DAG) from Conditional Independence maps where a major challenge is to minimize errors in the graph structure. This part of the chapter presents empirical evidence for best practice: to reduce false positive errors via the False Discovery Rate (FDR), and to identify appropriate parameter settings to improve the False Negative Reduction (FNR). In addition, several node ordering policies are investigated that transform the skeleton graph into a DAG (edges orienting rules), the results show that ordering nodes by strength of mutual information recovers a representative DAG in reasonable time, although a more accurate graph can be recovered using a random order of samples at the expense of increased computation time.

The CI-map methodology will then be used on the brain tumour data. The relevance of this dataset for CI-maps is the high association of specific metabolites with certain types of brain tumours, in such a way that the dominant associations using the proposed structure learning algorithm can be identified and visualized to aid interpretation. The CI-map recovered will differ depending on the type of brain tumour selected. Additionally, this study presents a good opportunity to apply bootstrapping methods to discover a robust set of metabolites that are the most predominant by tumour type.

The chapter is divided into two blocks, the first one tries to assess the parameters and policies in the PC-algorithm to build feasible CI-maps and BNs, using benchmark data where the true structure is known. The second block implements these procedures to brain tumour data.

The first part of this work continues the research developed in [87], where the aim is to present empirical evidence for best practice in setting three parameters optimizing: the policies of *False Discovery Rate* and *False Negative Reduction*, and the *effect of node ordering* when the edges are being oriented. This empirical research has conducted to the following contributions:

1. Analysis of structure finding based on PC algorithm and its dependence on design parameters e.g. data ordering.
2. Empirical parameter optimization of conditional independence maps removing dependence on design parameters.
3. Optimisation of False Discovery Rate (FDR) and False Negative Reduction (FNR) policies to avoid proliferation of False Positives.

The first point was mentioned in section 1.6 of the first chapter as one of the novel contributions of this thesis. The node ordering is further developed in section 5.4.2 of this chapter.

The second part of the chapter continues with a final procedure of bootstrapping the data for structure finding to address new data based on the parameters set-up of the PC algorithm. This procedure is applied to brain tumour data from MRS for analysing the metabolite dependences.

5.2 Background

This section describes three important parameters which must be taken into account in the structure learning when using the PC-algorithm, and then a bootstrapping method for selecting the most representative CI-map, assuming there is a target variable as a reference point.

5.2.1 False Discovery Rate

When multiple conditional independence tests are carried out, the control of False Discovery Rate (FDR) in the PC-algorithm is necessary. To understand the reasons, the basic mechanics of the PC-algorithm must first be explained:

The PC-algorithm performs conditional independence test between every pair of nodes, these test are conditional because assume that certain node values are given (observed).

The starting point of the algorithm considers that all nodes are dependent, i.e. a fully connected graph, and then the algorithm starts to perform multiple conditional independence test in successive stages, where in each stage the number of conditioned nodes are increased. The null hypothesis of all conditional independence tests is that they are independent. After each test where null hypothesis is not rejected, these nodes are considered independent and their edge is cut, meaning that this edge is no longer considered in the following conditional independence tests. Last point is critical to reduce the runtime, because the number of the independence test performed by the PC-algorithm grows exponentially with the possible combination of conditioned nodes if in the first stages enough nodes are not independent.

The stages of the PC-algorithm are described as follows: The first stage, the algorithm performs all combinations of independence test, non-conditioned. In the second stage, the algorithm starts to perform conditional tests adding only one conditioned node,

where the tests are repeated with different conditioned nodes. The next stage the algorithm repeats the conditional independence test but adding two conditioned nodes, and excluding those node combinations that were independent. And future stages continue in a similar way until all nodes are conditionally independent.

As explained before, the null hypothesis of the conditional independent test is that the nodes are independent, and the null hypothesis is rejected when the p-value of the test is lower than level of significance, α, therefore α controls the quantity of False Positives (the test indicates *dependence* when in reality there is not), the so-called type I errors. However, the more independence tests are made, the more likely False Positives (FP) are to occur by chance, this problem is known as the multiple comparisons problem. And because the PC-algorithm performs many independence tests on the same data, the FDR is needed to keep the same FP ratio on the multiple tests made. The FDR modifies α in such a way the expected proportion of FP remains constant.

The work [87] proposes three different variants of the FDR control policy: *Basic*, *interleaved* and *mini-FDR*. The FDR procedure is an extension of the results in [120].

- The default option is the *basic* FDR and is applied after PC algorithm convergence, a-posteriori.
- The *interleaved* variant is applied after each step of the PC algorithm, where a step implies one more node in the conditional independence tests. With respect to "a-posteriori" FDR, this procedure is interleaved within the steps of the PC algorithm.
- *Mini-FDR*, with respect to "a-posteriori" and "interleaved" FDR, is solely responsible for pruning edges in the structure.

5.2.2 False Negative Reduction

A False Negative (FN) implies a missing edge (independence) between two nodes (variables) when there is a true dependency present between the pair of nodes. The FNR policy, based on [89], avoids testing the hypothesis if the minimum power of the test is not at least $1-\beta$, with β being the false negative proportion. This condition establishes a maximum degrees of freedom threshold that depends on the sample size, β, the test level α, and a desired effect size w. In practice, β and α are usually pre-established standard values, and the effect size w has to be adjusted. This is the challenging part of the FNR, as the optimal w value is dependent on the data and its sample size, and therefore it is difficult to find a good value for unknown data.

The difficulties of the FNR policy appears in the following scenario: The process for finding the skeleton starts with a fully connected graph, FNR avoids some tests between nodes, and therefore the edges are kept, reducing the chance of a FN. But, at the same time, if too many edges are preserved because they are not tested, either due to a wrong parameter like w, or because some of these edges belong to true independent nodes and should be removed, it would create an excess of False Positives (FP). The key point resides on how to determine the appropriate w value for new data. This point will be investigated in the methodology section with benchmark data.

5.2.3 The effect of node ordering

As commented at the end of section 5.1, the PC algorithm solutions are sensitive to node ordering. The work in [87] addressed this problem by ordering the nodes by mutual information, and then testing the weakest nodes first. In such a way, the PC algorithm starts to prune the fully connected skeleton by the weakest nodes. Applying this policy the PC algorithm reconstructs the structure independently of the node order.

The next step is to orient the edges of the skeleton to produce a DAG, using a set of rules developed in [22, 23]. These rules are also sensitive to the node order, i.e. the order in which the edges are oriented affects the final DAG, this problem is even worse than the skeleton case because there is no unique solution.

By systematically following these rules, graph acyclicity is not guaranteed, thus after orienting an edge the acyclity must be checked, and if the acyclity is not preserved the last edge orientation is reversed. One of the complexities orienting an acyclic graph is that it has a tree structure, hence the edge orientation of the ancestors determine the edge structure of their descendants, and a mistake made in an ancestor edge is more punitive than a mistake in a descendant edge. The number of possible edge orientations grows non-linearly with the degree of descendants.

The theorem 3.7 from [85] states that two graphs with the same skeleton and the same set of V-structures (immoralities), are *I-equivalent* graphs. The group of *I-equivalent* graphs can be expressed as a Partially Directed Acyclic Graph (PDAG). However, the converse does not hold, graphs of the same *I-equivalent* group (and same skeleton) can have a different set of V-structures.

Therefore, knowing the skeleton and the V-structures set, only a PDAG can be unequivocally built. For building a DAG from the PDAG, one of the multiple solutions from the PDAG has to be chosen. The process of orienting edges depends on the node ordering

and taking into account that a graph with n nodes have $n!$ different node permutations, a couple of options are proposed to overcome this computational explosion:

- The first option is to once again order the nodes by mutual information, which gives two possibilities to orient the edges, by the weakest first (TWF) or the strongest first (TSF).
- The second option is to implement random node orders, build the DAGs and pick the solution with the best score. The score being the log-likelihood function with BIC regularization to penalize graph complexity.

The node strength is based on mutual information, and it measures the accumulated mutual information from one node with respect to the other nodes, computed by pairs.

In section 5.5.3, the effect of node ordering will be assessed measuring the score distributions of samples taken from the Insurance network.

5.2.4 CI-maps as feature selection

Sometimes, a CI-map can be built on a dataset with a target variable in mind, this target variable can belong to the dataset as a feature or as an class label, for instance class labels to be classified later. By analysing the CI-map associations (edges) in relation to the target variable, can be obtained which features (nodes) are most associated with the target variable, considering the first or even the second order edges as the influence neighbourhood of node dependence.

Building the CI-map using the PC-algorithm with the set-up developed in this chapter, a skeleton is obtained where the variables that are connected with our target variable are identified, but how robust is this CI-map? Are these variable associations the most representative from the information contained in the data? To test this a set of CI-maps can be created from bootstrapped data, i.e. re-sampling with replacement keeping the same size dataset. These new bootstrapped CI-maps should have a similar skeleton with little variations in the edges.

Analysing the frequency with which the target variable establishes its first and second order connections the nodes identified as having the most frequent associations can be used as an implicit measure of edge robustness. From the collection of CI-maps generated by data bootstrapping, a hierarchical filtering is proposed where the maps are filtered sequentially following the order of most frequent connections in relation to the target

variable, in such a way that at the end, the filtered CI-map is the most representative map of the collection.

With this CI-map, feature selection is achieved by only considering the features connected to the target variable. Depending on the purpose of the analysis, these can include first order or also the second order associations. This analysis is always made in relation to a target variable, if the target variable is changed, most probably the most representative CI-map will be different.

It is worth mentioning an issue related to the p-value of the independence tests and the significance of the variable associations, specifically to the ratio between the sample size and the degrees of freedom. Recalling that CI-maps require categorical variables, any continuous features have to be categorized, usually by quantiles. However, this affects the number of total categories and, in turn, affects the degrees of freedom of the independence tests, which are based on G^2 statistic. There is a rule of thumb where the G^2 statistic is not a suitable approximation of the χ^2 distribution if the ratio of the sample size and the degrees of freedom falls below 5, hence, it has to be expressed as follows:

$$\frac{\text{sample size}}{DoF} \geq 5 \tag{5.1}$$

Therefore, the degrees of freedom may need to be controlled by adjusting the number of categories for continuous features. Depending on the sample size, this can simply be achieved by controlling the quantiles or merging categorical data into larger groups.

5.3 Data description

5.3.1 Benchmark data for validation

For the benchmarking experiments, two well-known datasets have been used, namely: the Insurance and Alarm datasets, both available in the Bayesian Network Repository[1]. The Insurance dataset is a Bayesian network for evaluating car insurance risks [121]. It has 27 variables (nodes), 52 edges degree (arcs), 984 parameters, 5.19 average Markov blanket size, 3.85 average degree and 3 maximum in-degree. The structure is presented in figure 5.1.

[1] `http://www.cs.huji.ac.il/~galel/Repository/`

The "A Logical Alarm Reduction Mechanism" (ALARM) is a network designed dataset to provide alarms during patient monitoring in anaesthesia [122]. This data has 37 variables (nodes), 46 edges degree (arcs), 509 parameters, 3.51 average Markov blanket size, 2.49 average degree and 4 maximum in-degree. The structure is presented in figure 5.2.

Knowing beforehand the probabilities that describe these Bayesian Networks, it is possible to generate sample data of different sizes. In this work, sample sizes from 0.5k up to 100k observations have been used.

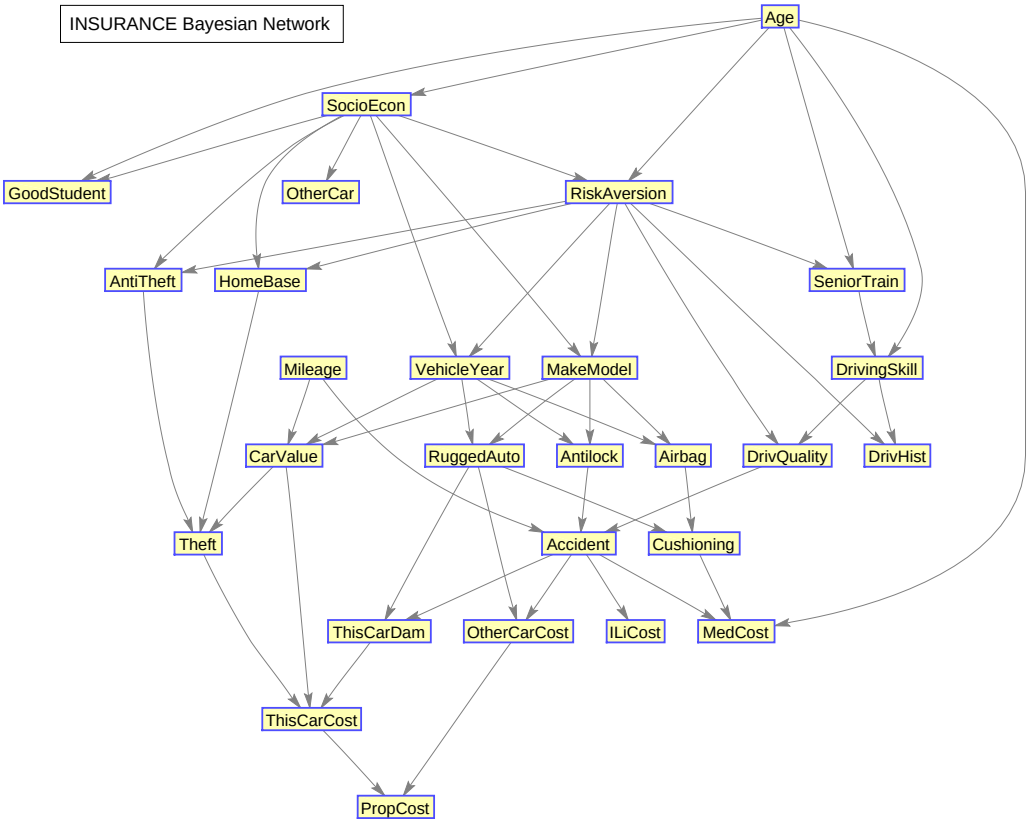


FIG. 5.1 – Insurance Bayesian network

5.3.2 Brain tumour data

The CI-map methodology has also been applied to brain tumour data based on MRS from 304 patients with different types of brain tumours. For this dataset, a tumour category is defined as the target variable, by applying the CI-maps methodology for feature selection, the main objective is to find associations between metabolites signal peaks and the tumour categories.

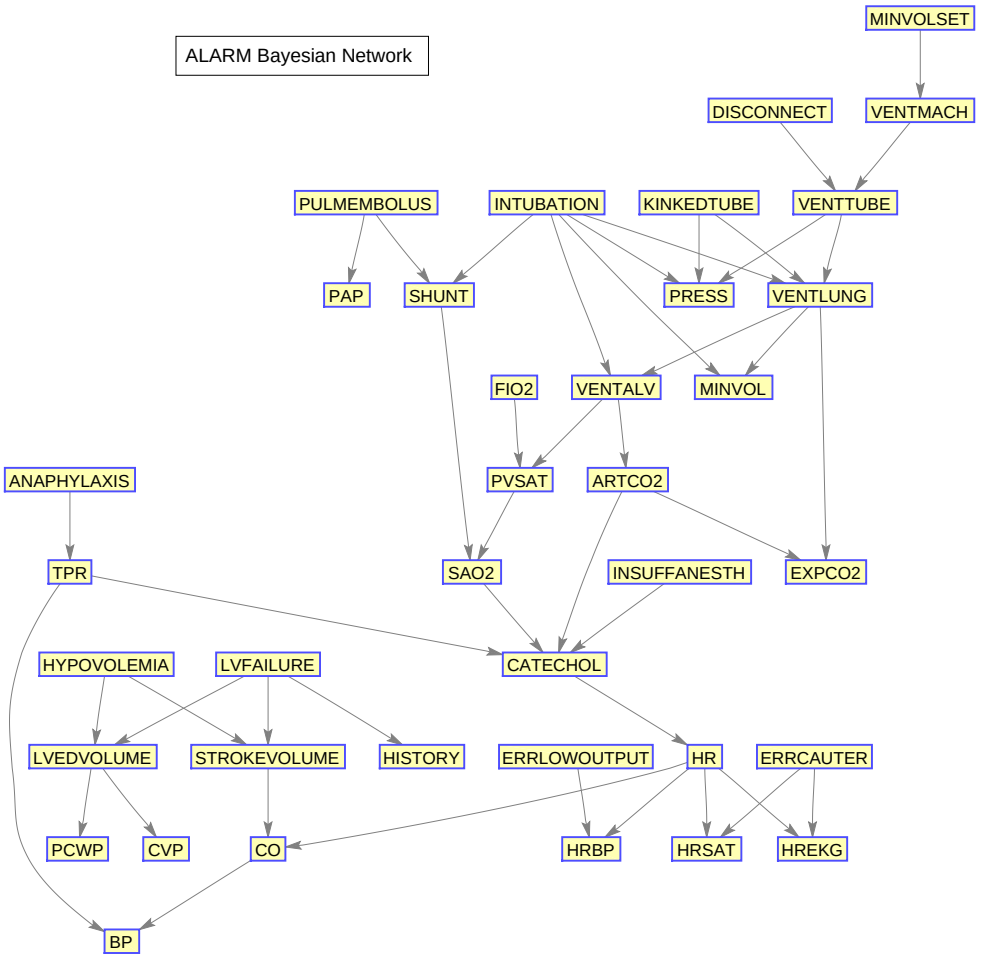


FIG. 5.2 – ALARM Bayesian network

The dataset used was extracted from INTERPRET, an international multi centre database [123] resulting from the INTERPRET European research project [124]. Class labelling was performed according to the World Health Organisation system for diagnosing brain tumours by histopathological analysis of a biopsy sample. These are single-voxel proton MRS (SV-^1H-MRS) data acquired at 1.5T. The spectra acquired at short echo time (20-32 ms) from brain tumour patients and healthy controls is being used. A list of them can be seen below (the brackets show the number of patients in each class):

- a2: Astrocytomas of WHO grade II (22 obs.)
- a3: Astrocytomas of WHO grade III (7 obs.)
- ab: Brain Abscesses (8 obs.)
- gl: Glioblastomas, giant cell glioblastomas and gliosarcomas (86 obs.)

- hb: Haemangioblastomas (5 obs.)
- ly: Brain Lymphomas (10 obs.)
- me: Brain metastases (38 obs.)
- mm: Meningiomas of WHO grade I (58 obs.)
- no: Normal brain tissue, white matter (22 obs.)
- oa: Oligoastrocytomas of WHO grade II (6 obs.)
- od: Oligodendrolgiomas of WHO grade II (7 obs.)
- pi: Pilocytic astrocytomas (WHO grade I) (3 obs.)
- pn: Primitive neuroectodermal tumours and medulloblastomas (9 obs.)
- ra: "Rare tumours", for more information about their pathologies, check the INTERPRET database (19 obs.)
- sc: Schwannoma (4 obs.)

MS spectroscopy provides insight into the biochemistry of tissue through a discrete signal in the frequency domain that reflects the relative abundance of several low molecular weight metabolites, lipids and macromolecules in the millimolar range of concentration. A total of 195 clinically relevant frequency intensity values measured in parts per million (ppm) were sample from each spectrum in the [4.24, 0.50] ppm interval. The following list describes a number of metabolites present in different types of brain tumours:

- Mobile lipids (ML). Found at 1.28 ppm. These MR-visible lipids are composed of triglycerides and cholesterol esters that accumulate in intracellular neutral lipid droplets. In human tumours mobile lipids are observed predominantly in high-grade brain tumours and in regions of necrosis [125].
- Lactate (Lac) / Lipids. Found at 1.31 ppm. Lactate can be observed as a doublet from the methyl group, at 1.31 ppm. However this spectral region is frequently complicated by the presence of lipid resonances [126].
- Alanine (Ala). Found at 1.46 ppm. Alanine has a doublet with centre at 1.46ppm. Increased alanine has been observed in vivo in meningiomas [127].
- N-Acetyl aspartate (NAA). Found at 2.01 ppm. Singlet at 2.01 ppm is the most prominent resonance of NAA.

- Creatine (Cr) / Phosphocreatine (PCr). Found at 3.03 ppm. Creatine and phosphocreatine (or creatine phosphate) can be observed as a prominent singlet resonance from their methyl-protons, at 3.03 ppm [126].
- Choline (Cho)-containing compounds. Found at 3.21 ppm. The choline signal is primarily observed as a prominent singlet at 3.21 ppm, which includes contributions from free choline, glycerophosphorylcholine, and phosphorylcholine, and it is often referred to as 'total choline' [126].
- Taurine (Tau). Found at 3.42 ppm. Taurine can be seen as two triplets at 3.25 and 3.42 ppm. For in vivo studies at lower field strengths, these resonances commonly overlap with the resonances from myo-inositol and choline.
- Glycine (Gly) / Myo-inositol (m-Ins). Found at 3.55 ppm. Glycine has two methylene-protons that co-resonate at 3.55 ppm. The glycine resonance overlaps with those of myo-inositol, making unambiguous observation of glycine not possible at shorter echo times [126].
- Glx group – with glutamate and glutamine (Glx). Found at 3.74 ppm. Glutamate (Glu) can be observed as a doublet-of-doublets centred at 3.74 ppm. Glutamine (Gln) is structurally similar to glutamate with two methylene groups and a methine group, and its coupling pattern is the same. A triplet from the methane proton resonates at 3.75 ppm [126].
- Alanine (Ala). Found at 3.77 ppm. Quartet centred at 3.77 ppm.

For this analysis, the tumour types are grouped as in [124], which includes the following classes: class 1: meningiomas, class 2: aggressive tumours (glioblastomas and brain metastasis), and class 3: low grade glials (astrocytomas grade II, oligodendrogliomas, and oligoastrocytomas). The normal parenchyma is also added to this set of classes as a class 4. The peak intensities (numerical variables) were categorized into 3 quantiles.

These tumour groups are later processed as four independent binary variables, where each one acts as target variable for the CI-map bootstrapping. Only one target is included in a CI-map to avoid the creation of spurious associations between tumour types which can mask the metabolites associations.

5.4 Methodology

The first part of this section is focused on the assessment of the PC-algorithm parameters and policies. The second part introduces and assesses the procedures applied to brain tumour data to build a feasible CI-maps and BN where the true answer is unknown.

5.4.1 Assessment of policies and parameters

First, the three options of the FDR policy are analysed to identify the procedure that produces the fewest skeleton errors in the Insurance dataset, then the FDR is also tested with the FNR policy with the appropriate parameters for the Insurance data, the idea is to analyse any synergies between both policies.

As described in Subsection 5.2.2, the FNR depends on the effect size, w. Therefore, the main objective is to determine, empirically, which is the range of w values that minimizes the total skeleton errors of the CI-map given the same sample size of the Insurance data. For this purpose, multiple CI-maps are retrieved by varying w. By plotting the False Positives, False Negatives and the total errors, the optimal w value that minimizes the total amount of skeleton errors can be identified.

The procedure to identify an appropriate value for w is then repeated for different datasets and sample sizes, where the variability of the appropriate w values against the sample size of the data is presented. The results are compared with those estimated in [89]. The latter analysis of the FNR policy is carried out by plotting the skeleton results for the Insurance dataset when the sample size is considerably larger than the sizes where a reasonably good w value can be estimated heuristically. These experiments are repeated without FNR policy, whereby only the FDR policy is considered.

5.4.2 Node ordering in Bayesian network's assessment

This subsection describes the methodology for assessing a BN using a BIC score as a function of the node order. Two different ordering of nodes based on mutual information are proposed, and the BN generated with these predefined orders are compared with those generated by a random node ordering. The main goal is to assess the score distributions as a function of the number of random node orders needed to generate BNs that outperform networks with predefined node orders.

The methodology consists of generating 100 samples of 25K observations of the Insurance dataset. Recalling that samples can be generated because the probabilistic model is known for this benchmark BN. The samples will produce 100 different CI-maps per data sample. Each CI-map provides a PDAG that can generate multiple compatible DAGs which represent the BN, where the generated BNs depend on the node order on which the edges are oriented. For the experiment, the 100 BNs are scored with regards to the predefined orders by mutual information, the weakest first (TWF) and the strongest first (TSF). Then several sets of BN are generated by random node order, selecting the ones with best and worst scores, this is applied to the 100 CI-maps, creating a score

distribution. The sets are composed of 1, 25 and 100 different BNs with random node ordering. The three distributions are compared in the three-random-set scenarios.

The main idea is to have a decent baseline of a BN score with a predefined node order, later it will be seen that the TSF is the best one, and then run random node iterations until the BN score outperforms the best old score. The number of iterations depend on the computation time available. This approach provides a non-arbitrary baseline result for a realistic assessment of the performance of the respective policies.

5.4.3 Most representative CI-map with bootstrapping

In this subsection, the methodology for highlighting the strongest features associated with the type of brain tumour is presented. The methodology starts with the pre-processing of the brain tumour data, where its features are numerical variables obtained from the maximum signal of the MRS within the spectral range in ppm associated with each metabolite. The metabolites are described in section 5.3.2. The signal features are discretized into categorical variables using 3 quantiles.

The target variable is an class label with the information of the tissue type per case (patient or control). In total there are seven different types of tissue (brain tumours and normal brain). They are grouped into four categories: lowgrade, aggressive, meningioma and normal, as explained previously. For the input of the CI-map, each tissue type category is considered as a binary variable isolated from the rest of categories. Only one tissue type category is included per CI-map in order to obtain its feature associations without interferences. Therefore, the process will be repeated four times, one per tissue category.

Next step is to generate 400 bootstrapped data creating 400 CI-maps is not computationally expensive for this size of data and is enough to obtain robust results. The results are recorded in an accumulative list of the first and second order associations connected to the tissue category. From this list, two histograms are constructed in order to visualize the most frequent associations for the first and second order connections of the target variable.

Inspecting the histograms provides qualitative insights about the structure of the CI-maps, for instance, if two nodes have complementary percentages between the first and second order edges, they probably are swapping positions in the edges. If in the first order histogram there are few nodes but with high percentages, this means the connections are very significant and most of the CI-maps are quite similar. On the other hand, if there is a number of nodes connected to the target variable, this means that there are

many weak connections where edge variation is considerable, probably indicating that there are a number of weak connections and these nodes can appear as 1, 2 or 3 order connections or no have connection at all (which means independence).

The next step in the procedure is to apply the hierarchical filtering, starting with the first order connections and selecting the most frequently connected node, then the second order and so on, until there is only one remaining CI-map or the rest are equal. In fact, after the third order connection the filter can be discarded, and just select one random node of the remaining maps. Nodes with a frequency less than 10% are ignored to avoid maps with low prevalence.

The selected CI-map is the most representative in terms of frequency that the nodes appear in the set of CI-maps. This procedure is a robust method whereby the target variable connections represent the strongest, most reproducible dependences. The CI-map can then also be used to build a BN following the methodology described in Subsection 5.4.2. The CI-maps identified using this approach are presented in the results for the chapter from section5.5.4 onwards.

5.5 Results

The influence of the FDR, the FNR and the node ordering in two known Insurance and Alarm networks are now presented. The effects are assessed by counting the errors in the skeleton compared to the true structure. An error is considered as having: too many edges (FP), missing edges (FN), and the wrong edge direction in the case of a DAG. The Bayesian networks formed by DAGs are also evaluated using a log-likelihood score function with BIC as a regularization term.

5.5.1 False Discovery Rate

Using the Insurance network, a comparison table has been created to measure the skeleton errors with six possible combinations of FDR and FNR. Table 5.1 shows the skeleton errors for a sample of 500 observations of the Insurance dataset, for each of the FDR control policies. The FNR setup has an effect size of $w = 0.25$, and power of $\beta = 0.05$. The weakest first (TWF) node ordering is enabled. The table shows an improvement of 5 fewer False Positives (FP) when FNR is activated, but no significant changes with reference to the FDR. Therefore, the default policy, *basic FDR*, should be the recommended policy just because of the simplicity of the code.

TABLE 5.1 – Averaged skeleton errors of 10 samples of Insurance data with 500 observations each

SETUP	*FN*	*FP*	*TOTAL*
FDR basic	3.1 ± 1.2	25.3 ± 1.5	28.4 ± 2.5
FDR interleaved	3.0 ± 1.5	25.2 ± 1.5	28.2 ± 2.8
FDR mini	2.8 ± 0.9	25.0 ± 1.6	27.8 ± 2.4
FDR basic + FNR	3.3 ± 0.9	20.6 ± 1.6	23.9 ± 1.9
FDR interleaved + FNR	3.4 ± 1.3	20.4 ± 1.4	23.8 ± 2.4
FDR mini + FNR	3.4 ± 1.1	20.4 ± 1.4	23.8 ± 2.3

These results are compared with the R package called PCALG [128], using the same 10 samples of the Insurance dataset with 500 observations. Table 5.2 shows the skeleton errors of three structure finding algorithms from the PCALG package, the FCI algorithm [8], and two variants of PC-algorithm [92], called *special* and *relaxed*, where the former tests that no additional V-structures are needed if a DAG is built, and the latter does not apply this condition.

TABLE 5.2 – PCALG R package Insurance CI-map comparison

PCALG algorithms	Skeleton errors	
	500 obs.	5000 obs.
FCI	26.6 ± 1.1	24.7 ± 1.5
PC special	25.7 ± 1.0	22.6 ± 2.4
PC relaxed	25.7 ± 1.0	23.2 ± 2.4

The results are quite similar, slightly better than the algorithm with only FDR, but slightly worse when FDR and FNR are activated. The purpose of this comparison is to check if the results are approximately similar and consistent.

5.5.2 False Negative Reduction

Figure 5.3 shows the averaged skeleton errors of 10 different samples of Insurance dataset with 500 observations. This is to illustrate the pattern of the structure errors with reference to the effect size w, the observed behaviour is similar for different sample sizes but displaced to the left (same peaks but with lower values of effect size).

A number of insights can be gained from inspection of this figure:

- FN errors decrease as the effect size decreases until some point where all tests are avoided and the graph becomes fully connected. Beyond this point FNR is disabled.
- Similarly where the FN errors start to decrease, the FP errors start to increase until reaching the fully connected graph where the FP errors are at a maximum.

- There exists an effect size window, δw, where the trade-off between FP vs FN is acceptable, and the total error decreases. This δw depends on the dataset and the sample size.

- For this example of 500 observations, one may observe the optimal effect size is $w_{min} = 0.24$, improving the average total error by 4.7. This improvement diminishes as the sample size increases, being 4.7, 3.6, 3.3, and 2.3 skeleton errors for 0.5K, 1K, 5K and 10K samples, respectively. In the best case, an improvement of approximately 5 skeleton errors does not warrant the high risk of selecting a bad effect size for new data.

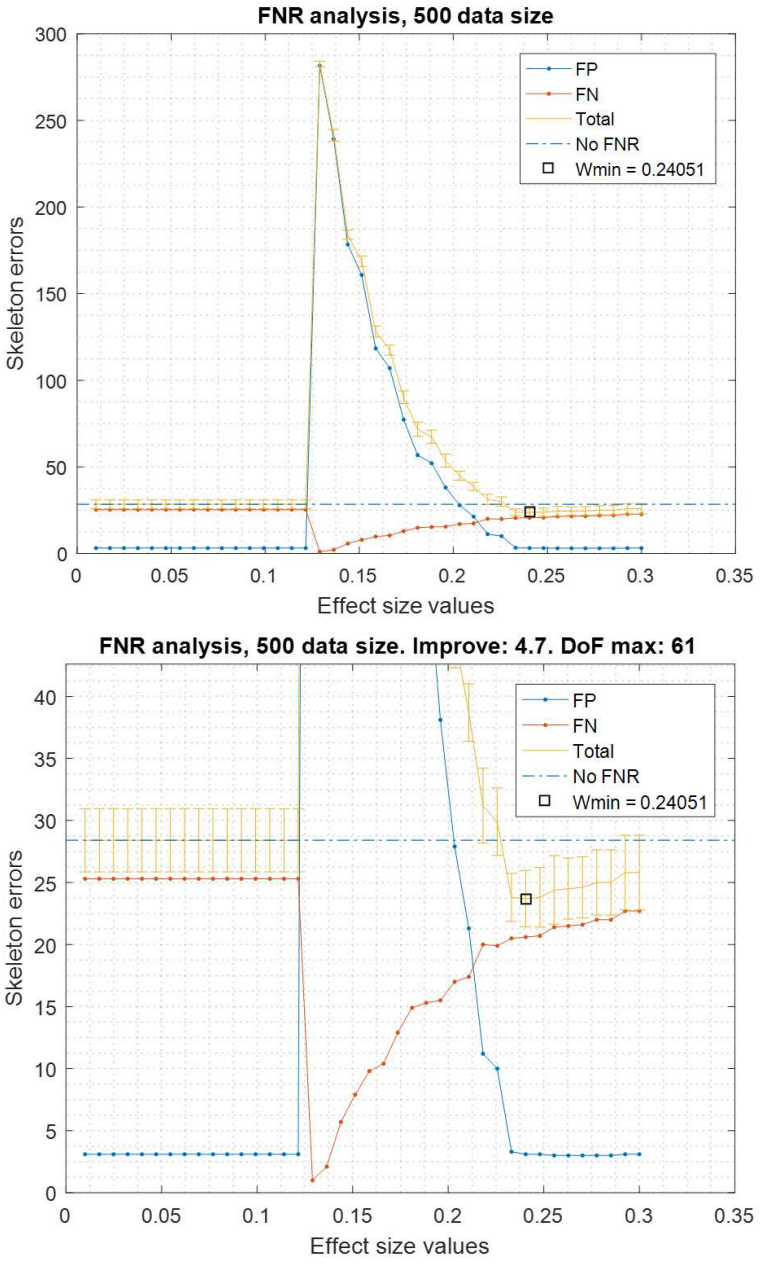


FIG. 5.3 – Averaged structure errors for Insurance BN. Zoom in critical point

Figure 5.4 shows the optimal effect sizes, w, found through empirical tests with Insurance (red) and Alarm (black) networks. The procedure is similar to that presented in figure 5.3, scanning several effect size values and selecting the one that minimizes the total skeleton errors. Additionally, in the same plot one may observe in blue the effect size suggested by [89], in that work the optimal effect size is obtained from a random sample of Insurance and Mildew networks using a cross-validation at each sample size. Also a bandwidth of $\Delta w \approx 0.15$ is bounded, with the lower bound $w_{LB} \approx w_i - 7.5$ an unconstrained skeleton, and the upper bound $w_{UB} \approx w_i + 7.5$ a rule of thumb; the rule of thumb states that the independence test (G2-test) is reliable if there are five or more instances per degree of freedom of the test. If the test is not reliable, PC-algorithm defaults decision to include the edge in the skeleton.

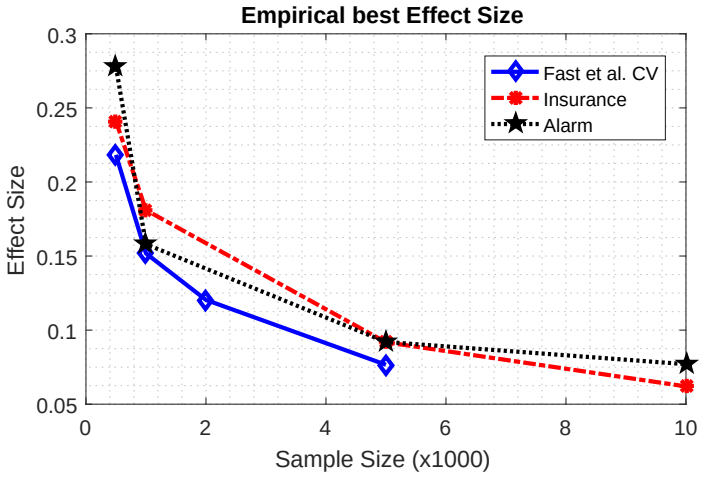


FIG. 5.4 – Best empirical values for effect size parameter

Using the suggested effect size for a 500 sample size in the figure 5.4, a $w = 0.22$ should be obtained. If that value is taken, figure 5.3, the errors would be higher than if FNR were disabled. Hence, FNR is not recommended for new datasets or if the effect size cannot be selected optimally.

Similarly, figure 5.5 shows how a fixed effect size w affects the skeleton errors when the data size increases. When FNR is activated, an increasing FP error is observed when the data size increases. If FNR is deactivated a significant FP reduction is observed, these plots show the effect size has to be carefully adjusted as a function of the sample size. Theoretically, the PC-algorithm would converge to the true structure if the whole population data were available. However, in a realistic case scenario, the skeleton errors asymptotically tend to a non zero value, ≈ 7.5. In this analysis, the errors for the DAG have a different source, it is related to the lack of a unique DAG solution given the skeleton. Due to the application of the orienting rules of [22, 23], it does not unequivocally solve the DAG, and this is why other works use the Partially Directed Acyclic Graphs (PDAGs).

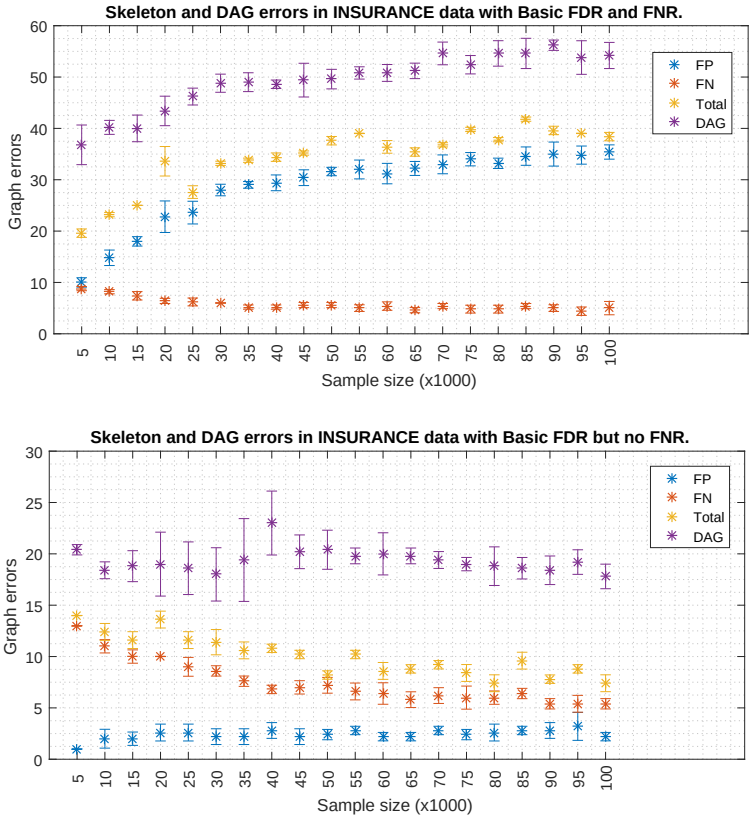


FIG. 5.5 – Skeleton and DAG errors vs sample size and FNR for Insurance data

5.5.3 The effect of node ordering

Now the score distributions of a sampled Insurance network are presented as a function of the node ordering when the DAG is created (edge orientation) from the skeleton built with the PC-algorithm.

Figure 5.6 shows the density functions of the BIC score, using a non-parametric kernel-smoothing distribution for 100 samples of the Insurance dataset with 25K observations and 27 nodes. In each plot, there are four distributions with TWF and TSF node order, and the best and worst solutions obtained by random ordering. There are three scenarios: only 1 random order (being only 1, worst = best), 25 random samples, and 100 random samples.

The graphs show that the TSF node order policy is the best option when there are limitations in the number of repetitions to build new DAGs based on random node orders. Such limitations may be, for instance, due to a computational time constraint for a large data set, or when the graph has too many nodes and a large number of repetitions is needed to extract a representative sample of the node order permutations. In contrast, the score distribution of the TWF node order policy is centred in a region

of lower scores and presents a long left tail. This tail can produce a very poor DAG score depending on the sampled data. However, the tail of TSF distribution is shorter and it defines a lower bound for the DAG score.

Intuitively, the reason why TSF has a better performance than TWF is because TSF starts orienting the strongest edges first, like in a tree structure the strongest edges being the biggest branches, and leaving the orientation of the weaker edges (branches) for the last steps of the algorithm. This effect is just the opposite in the CI-map pruning process (edges cut), where the performance improves if the fully connected graph is being pruned starting from the weakest edges.

However, when creating a new DAG sampling node order permutation does not require an excessive computational time, the best option to obtain a faithful DAG (highest score) is the random sampling option. The basic idea is to repeat the random samplings as much as reasonably possible, and pick the solution with the best score. In this example with 25K observations, the sample is considered that represents the whole population for the insurance dataset quite well. Decreasing the number of observations will produce more noisy distributions.

5.5.4 Brain tumour results

This subsection shows the results of the CI-maps applied to the four target variables based on tissue type categories with reference to the relevant brain tumour metabolites. The results are in the following order: normal, low-grade, aggressive and meningioma. For each target variable three figures are presented: 1) the association histograms in relation to the target variable, obtained from the bootstrapped CI-maps. 2) The most representative CI-map based on hierarchical filtering, and 3) the BN derived from the CI-map using the node order assessment, starting with TSF node order and improving the BIC score with random iterations.

In all CI-maps the edges include the mutual information, the significance test is set to a p-value of 0.05, and the bootstrapping has 400 iterations.

5.5.4.1 Tumour category: Normal tissue

As an example, in this first case figure 5.7 the original CI-map created without bootstrapping is shown, just so the original observations can be compared with the CI-maps obtained from the filtered bootstrapped CI-maps in figure 5.10.

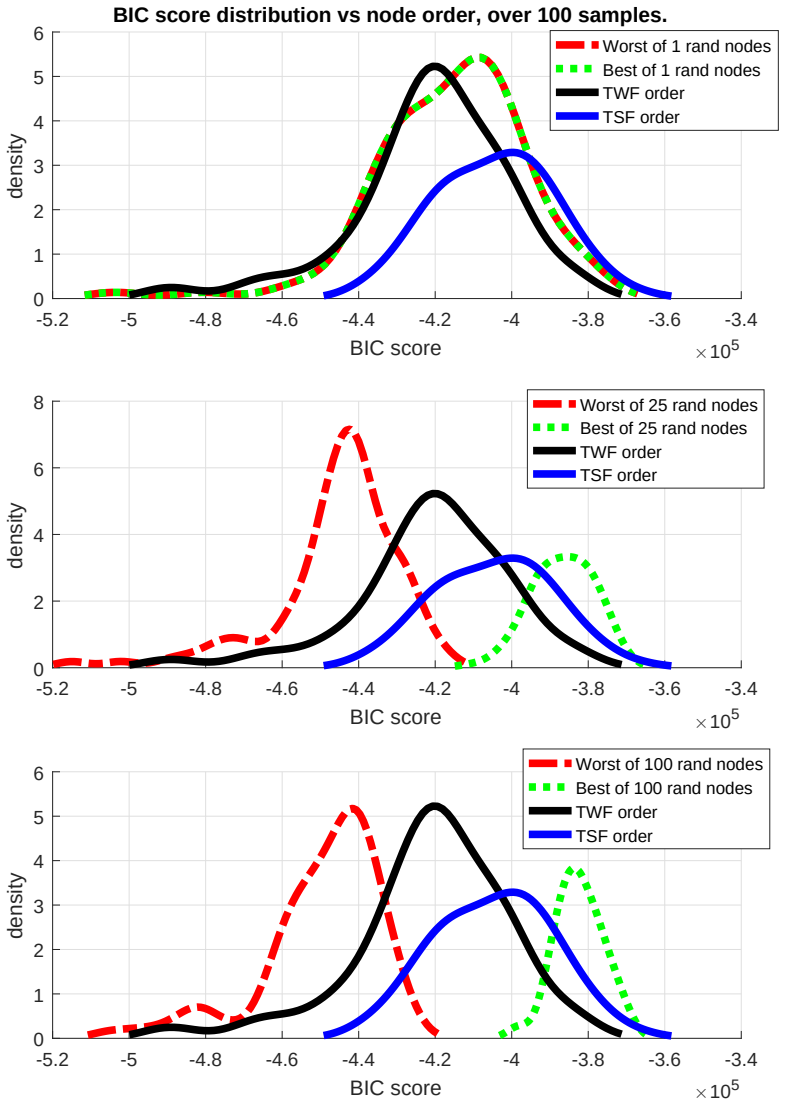


FIG. 5.6 – BIC score of node order distributions for Insurance network

In figures 5.8 and 5.9 are the associations histograms in relation to the tumour category (normal).

Figure 5.10 shows the hierarchical filtered CI-map, which is the most representative CI-map of the bootstrapped data with respect to the most frequent node associations in relation to the tumour category (normal).

Figure 5.11 shows the BN built from the bootstrapped CI-map applying the node order methodology to obtain the best BIC score.

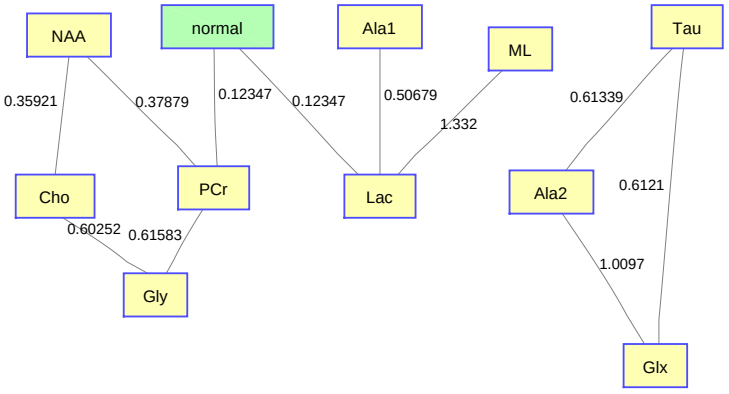


FIG. 5.7 – BT normal original CI-map

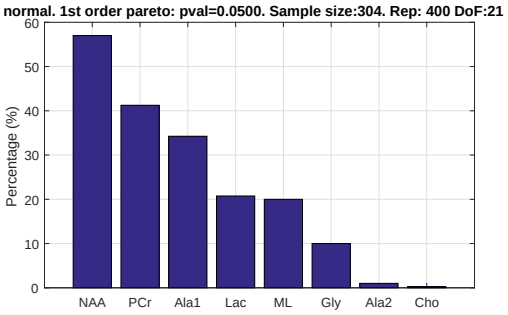


FIG. 5.8 – BT normal 1st order hist.

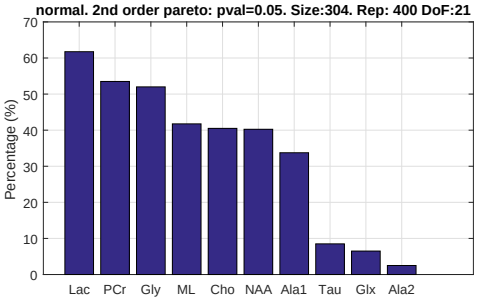


FIG. 5.9 – BT normal 2nd order hist.

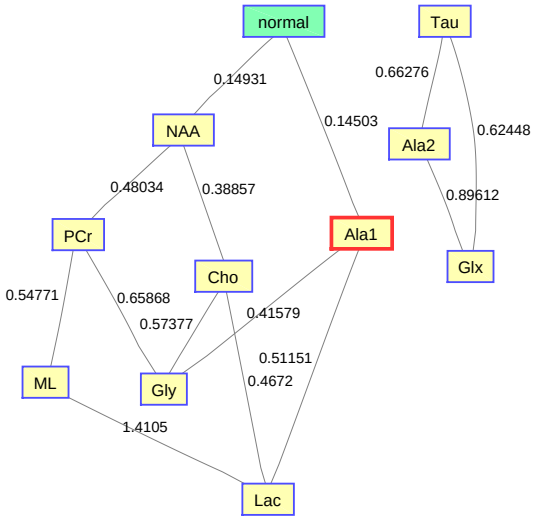


FIG. 5.10 – BT normal bootstrapped CI-map

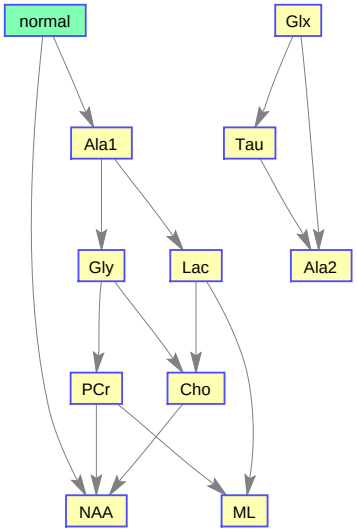


FIG. 5.11 – BT normal BN

5.5.4.2 Tumour category: Low-grade

In figures 5.12 and 5.13 there are the associations histograms in relation to the tumour category (lowgrade).

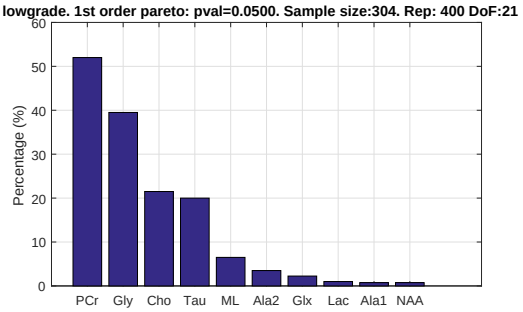


FIG. 5.12 – BT lowgrade 1st order hist.

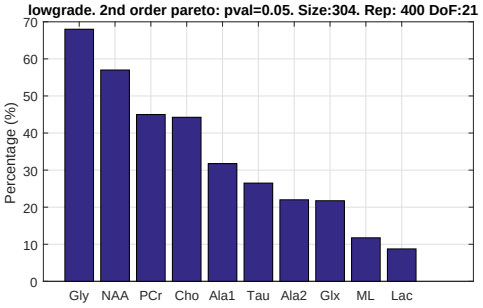


FIG. 5.13 – BT lowgrade 2nd order hist.

Figure 5.14 shows the hierarchical filtered CI-map, which is the most representative CI-map of the bootstrapped data due to respect the most frequent node associations in relation to the tumour category (lowgrade).

Figure 5.15 shows the BN built from the bootstrapped CI-map applying the node order methodology to obtain the best BIC score.

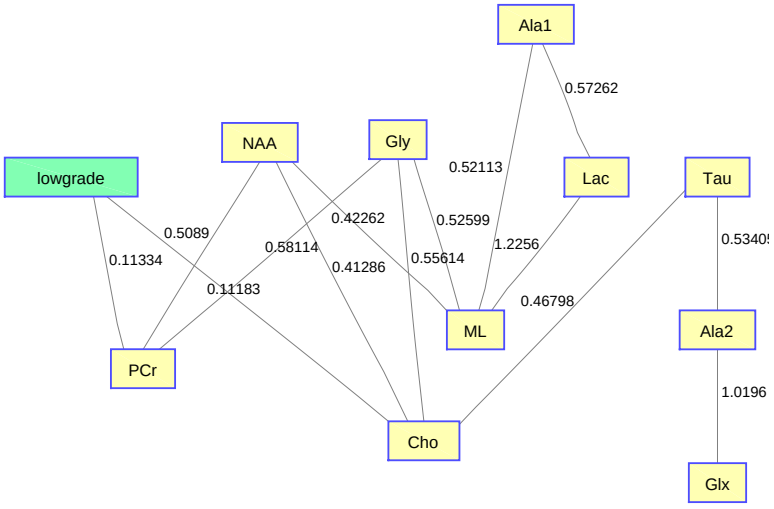


FIG. 5.14 – BT lowgrade bootstrapped CI-map

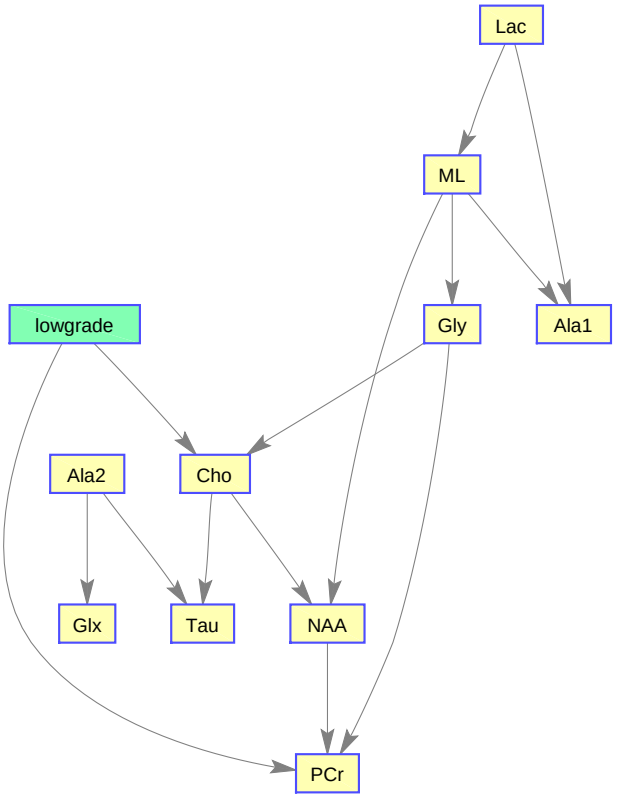


FIG. 5.15 – BT lowgrade BN

5.5.4.3 Tumour category: Aggressive

In figures 5.16 and 5.17 there are the associations histograms in relation to the tumour category (aggressive).

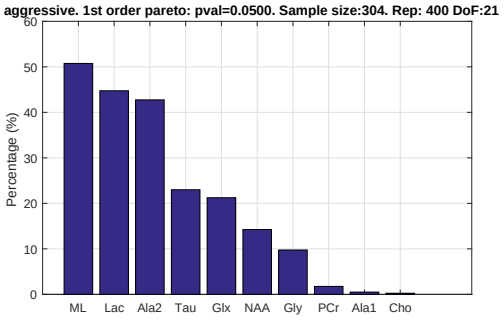


FIG. 5.16 – BT aggressive 1st order hist.

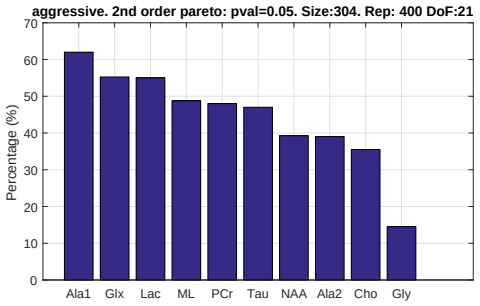


FIG. 5.17 – BT aggressive 2nd order hist.

Figure 5.18 shows the hierarchical filtered CI-map, which is the most representative CI-map of the bootstrapped data due to respect the most frequent node associations in relation to the tumour category (aggressive).

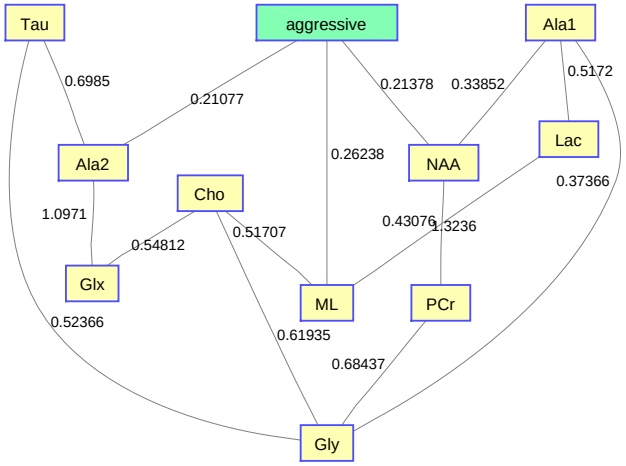


FIG. 5.18 – BT aggressive bootstrapped CI-map

Figure 5.19 shows the BN built from the bootstrapped CI-map applying the node order methodology to obtain the best BIC score.

5.5.4.4 Tumour category: Meningioma

In figures 5.20 and 5.21 there are the associations histograms in relation to the tumour category (meningioma).

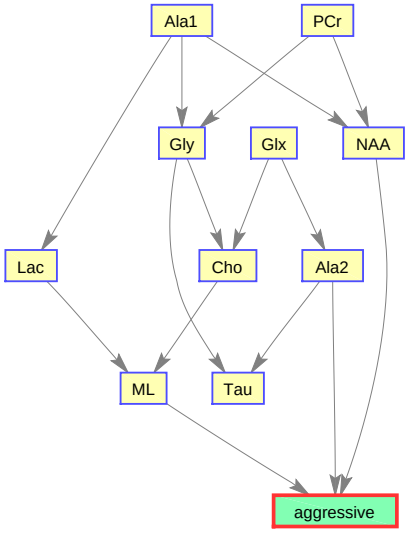


FIG. 5.19 – BT aggressive BN

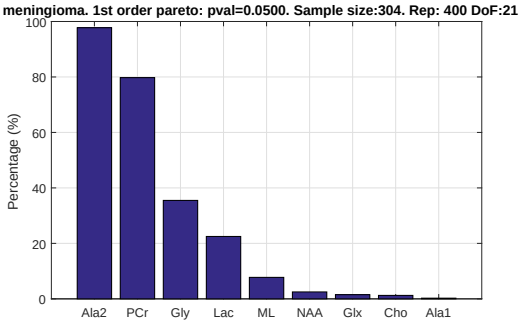


FIG. 5.20 – BT meningioma 1st order hist.

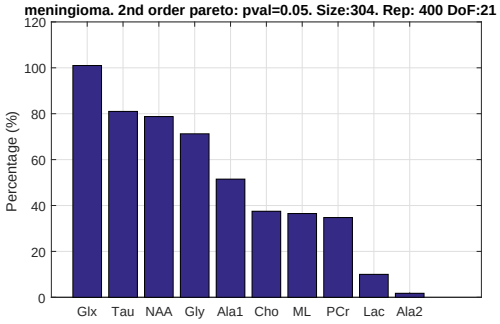


FIG. 5.21 – BT meningioma 2nd order hist.

Figure 5.22 shows the hierarchical filtered CI-map, which is the most representative CI-map of the bootstrapped data due to respect the most frequent node associations in relation to the tumour category (meningioma).

Figure 5.23 shows the BN built from the bootstrapped CI-map applying the node order methodology to obtain the best BIC score.

5.6 Discussion

5.6.1 CI-map methodology

With regard to the methodology and policies developed for applying the PC-algorithm to build CI-maps, based on these results the following conclusions can be made:

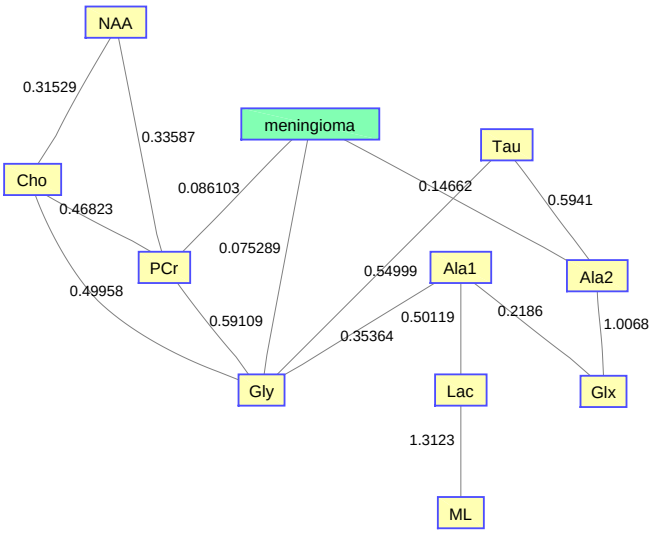


FIG. 5.22 – BT meningioma bootstrapped CI-map

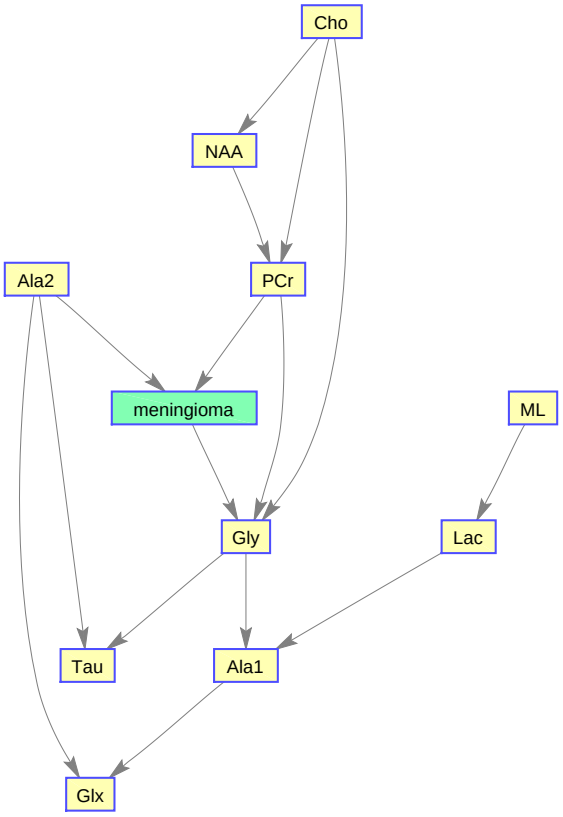


FIG. 5.23 – BT meningioma BN

- In terms of FDR policy, all the experiments performed in section 5.5.1 show no significant differences in the structure errors between any of the three FDR policies proposed in [87], without taking into account the FNR policy, which is safer not to activate, discussed in the next point.

- The FNR policy is able to reduce some FP in the CI-maps 5.1, but it requires a very fine adjustment of its parameters (effect size, w). For unknown data, which is most cases, the proper parameters are unknown, and a bad parameters adjustment can produce a high proliferation of FP as seen in 5.3. The FNR parameters also depend on the sample size, which complicates matters even more for its proper adjustment. For that reason, it is safer not to activate the FNR policy.

- When a DAG is created from a CI-map, the node order for edge orienting influences the BN outcome based on the BIC score. Figure 5.6 shows the BIC score distributions, where pre-ordering the nodes by strength of mutual information warrants a good baseline BIC score. However, it can be outperformed by BN with random node ordering, where it is necessary to perform as many iterations as practical, a 100 iterations being a reasonable number to obtain a significantly better score distribution.

- If there is a target variable, bootstrapping methods can be applied to obtain a robust identification of the most frequent edges connected with the target variable. From the CI-maps generated a hierarchically filter is applied to select the most reproducible CI-map, in other words, the CI-map most representative of the generated sample. From the CI-map a BN following can be constructed following the previous point.

5.6.2 Brain tumour CI-maps

In general, the metabolites associations with the tumour categories agree with the expected dependences described in the metabolites literature, see notes in 5.3.2. Analysing each category in more detail:

- Normal tissue: Sorting in decreasing order, the associations with highest prevalence are NAA, PCr and Ala1, with 58%, 42% and 35% respectively. The N-Acetyl aspartate (NAA) association is the expected result, being an abundant brain metabolite, predominant in normal brain tissue. Interestingly, the CI-map shows that there are other metabolites completely disconnected from normal tissue, like Tau, Ala2 and Glx.

- Low-grade tumours: Sorting in decreasing order, the associations with highest prevalence are PCr, Gly, Cho and Tau, with 52%, 39%, 22% and 20%. Here, the associations differ from normal tissues with the presence of both, PCr and Gly, where Gly was absent in normal and NAA is absent in low-grade.

- Aggressive: Sorting in decreasing order, the associations with highest prevalence are ML, Lac and Ala2, with 52%, 45% and 43%. According to the literature [129], the lactates and lipids (Lac) are associated with necrotic cells. In the selected CI-map, Lac is a second order connection with aggressive tumours through ML and Ala1.

- Meningioma: Sorting in decreasing order, the associations with highest prevalence are Ala2 and PCr with 98% and 80%. In this case, Alanine and creatine are clear indicators of meningiomas, which are located around meninges. Although this tumour is benign, it is important to discriminate it from other aggressive tumours, in order to promote accurate diagnosis, which will lead to more appropriate treatment and prognosis.

The most important associations extracted from the prevalence histograms can be summarized in the following figure 5.24. The solid arrows connects the first order connections, the widest arrows indicate the predominant association, and the dashed arrows indicate second order associations.

5.7 Conclusion

The focus of this chapter is on finding the best setup for structure finding stabilization based on the PC algorithm. With these empirical results, it can be concluded that any of the FDR policies can be used to control the FP errors; however, the basic FDR has been selected because of its simpler implementation. FNR policy is focused on decreasing the FN errors, but it is recommended not to activate when the optimal values of the effect size are unknown, which is the usual case for new data. The *effect size* parameter in FNR policy can produce a proliferation of FP errors if the parameter is not well adjusted to the data. Finally, to build a DAG from the skeleton, the node in which the edges are oriented affects the final DAG. Ordering the nodes by mutual information is proposed followed by orientating by the strongest first (TSF) order, which provided the best BIC score compared with a random ordering distribution. However, if the DAG is generated multiple times by sampling random node orders, the best solution of multiple random node orders outperforms TSF. The higher the number of samples the better BIC score is obtained, the price for this preferred approach is an increase in computational cost.

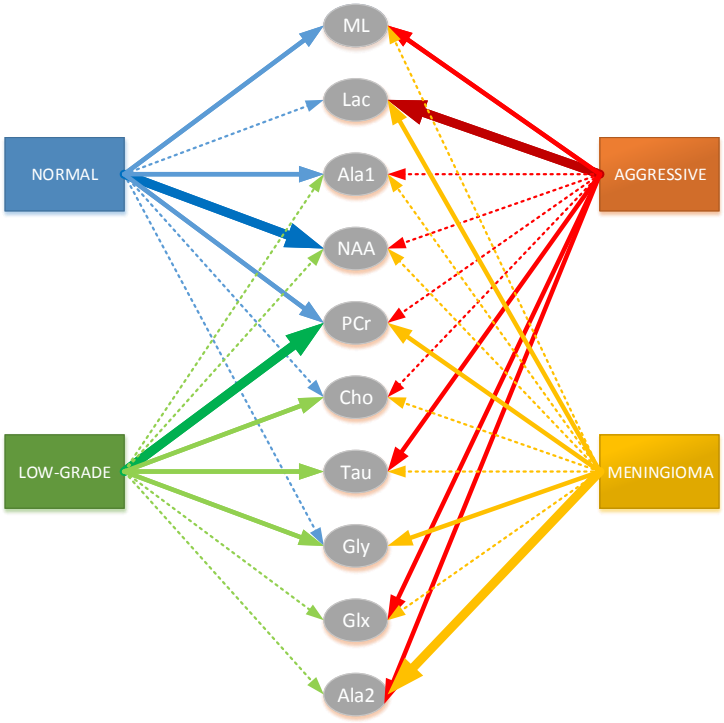


FIG. 5.24 – BT summary tumour vs metabolites associations

When investigating connections to a target variable the CI-maps can be used as feature selection, where using bootstrapping techniques robust results can be obtained with regard to identifying the highest edge prevalences with reference to the target variable. Apart from the variable associations, this method can be used to select the most representative CI-map from the bootstrapped CI-maps, subsequently to be transformed into a DAG following the procedure described in the first part of the work.

The CI-map methodology has been applied to brain tumour data based on MRS, producing consistent results about the metabolites associated with different tumour types that agree with the literature. With the bootstrapped CI-maps, histograms of the first and second order edges have been built, reflecting the prevalence of each association. Figure 5.24 summarizes, qualitatively, the most important metabolite associations.

Chapter 6

Complete framework and its application to retail and music data

This chapter analyses the procedure of the full framework using two case studies as examples. In the pipeline described in figure 1.1, step 6 presented a decision point between two paths where a clustering method had to be chosen to find clusters in the Fisher manifold. The objective of this chapter is to present two examples in which the two paths have been followed, and where both clustering methods will be compared. The idea is to extract insights about which clustering path is better depending on the manifold distribution.

The chapter is organized in three blocks: the first block, Section 6.1, computes the Fisher manifold of the two case studies and performs the two clustering methods on each case study comparing the results. The second block, Section 6.2 describes in detail the Fisher manifolds analysis based on customer shopping baskets but produced with different class labels. In addition for this case study, Bayesian networks are built from the shopping baskets grouped by the cluster labels. The section includes an introduction about the application of this methodology to shopping baskets, data preprocessing and feature selection, also different manifolds are built changing the class labels or stratifying the input data by other class labels with the objective of finding different customer segmentation. The last block, Section 6.3, builds a Fisher manifold based on spectral features of music data, with the objective of finding additional similarities (clusters) within songs that are predefined by genre labels. In this case study the PQC has been used to cluster the Fisher manifold because it presented strong density variations. On the other hand, the part of Bayesian networks is not applied here because there is no

interest in looking for feature relationships between the music spectral features. Finally, the chapter ends with a final conclusion.

6.1 Procedure of pipeline methodology

The overview about the pipeline methodology was depicted in figure 1.1. In this section is detailed the algorithm 2 which shows the procedure for obtaining the Fisher manifold, putting together the equations seen in chapter 3. The idea is to summarize the methodology, describing step by step until obtaining the adjacency matrix that defines the global pairwise distance of the Fisher manifold. According to the figure 1.1, the algorithm 2 describes the steps from 1 to 5.

Next is step 6, which uses the output of the algorithm 2, $\mathbf{D}_{manifold}$, to embed the Fisher manifold in a Euclidean space with the information of its pairwise distances. In Section 3.2.4.2 it is recommended to use cMDS because it preserves global distances better than Sammon mapping.

The goal of step 6 is to visualize the Fisher manifold distribution to help choose the type of clustering to be applied in the next step, symbolized as a decision point in the figure 1.1. This is discussed in next subsection.

6.1.1 Comparison of the two clustering methods

The pipeline figure 1.1 shows two possible paths for clustering the Fisher manifold, one path is focused on segmentation and the another on density discrimination.

The steps 7 and 8 form the path of segmentation, where the clustering task is performed through a network with community detection algorithms. The network is based on the distance adjacency matrix from step 5, $\mathbf{D}_{manifold}$, where the network is obtained after transforming the matrix into a similarity network with a Gaussian kernel tuned by a length scale parameter described in Section 3.2.3.

Alternatively, steps 9 and 10 are based on the Euclidean embedding of the Riemannian manifold, with this Euclidean space projective methods can be applied to analyse the manifold density distribution. One the techniques proposed in this work is the probabilistic Quantum Clustering, which is specialized in density discrimination.

The issue in question is which path to choose, the answer is that both paths can be valid. Although the manifold distribution can favour a technique, both paths are not

Algorithm 2 Procedure of obtaining Fisher Information manifold

Input data: collection of N observations with k features, $\mathbf{x} \in \mathbb{R}^k$
Class labels: collection of N labels of J classes, c_j with $j \in [1, J]$

$\mathbf{X} \leftarrow$ Standardized Input data ▷ *Z-score is recommended for MLP*
$\mathbf{C} \leftarrow$ Class labels
$P(\mathbf{C}|\mathbf{X}) \leftarrow$ Probabilistic discriminative model ▷ *MLP is the model used*
$\mathbf{FI}(\mathbf{x}) \leftarrow$ Fisher Information metric with respect to $\mathbf{x}$ ▷ *For generic probabilistic model see eq. 3.8, for MLP see eq. 3.14*

procedure LocalPairwiseDistances($\mathbf{X}$)
    $\mathbf{D}_{SL} \leftarrow$ Matrix of zeros of size (N,N)
    for all $i, j \in [1, N]$ with $i < j$ **do** ▷ *Spark parallelizes the list of pairs(i, j)*
        $d(\mathbf{x}_i, \mathbf{x}_j)^2 \leftarrow$ Distance SL eq. 3.18 ▷ *Distance derived from eq. 3.16*
        $\mathbf{D}_{SL}(i, j) \leftarrow d(\mathbf{x}_i, \mathbf{x}_j)$
        $\mathbf{D}_{SL}(j, i) \leftarrow \mathbf{D}_{SL}(i, j)$ ▷ This matrix is positive and symmetric
    end for
    return $\mathbf{D}_{SL}$
end procedure

procedure GlobalDistances($\mathbf{D}_{SL}$)
    if $\mathbf{D}_{SL}$ fits in memory **then** ▷ Non-distributed computing
        Implementation: APSP Floyd-Warshall algorithm in Matlab
        $\mathbf{D}_{geo} \leftarrow$ APSP($\mathbf{D}_{SL}$)
    else ▷ Distributed computing
        Implementation: SSSP Dijkstra algorithm in Spark
        $k \leftarrow$ Random list of K observations, with $1 \leq k \leq N$
        for all $k \in K$ **do**
            $\mathbf{D}_{geo}(k, :) \leftarrow$ SSSP$(k, \mathbf{D}_{SL})$ ▷ *Row vector with N shortest paths from k to the rest of nodes*
            $\mathbf{D}_{geo}(k, :) \leftarrow \mathbf{D}_{geo}(:, k)$ ▷ This matrix is positive and symmetric
        end for
        for all $i, j \in [1, N]$ with $i < j$ **do**
            $d_{aux} \leftarrow \infty$ ▷ *Auxiliary variable to iterate the k' distances*
            for all $k' \in K$ **do**
                $d_{temp} \leftarrow \mathbf{D}_{geo}(i, k') + \mathbf{D}_{geo}(k', j)$ ▷ *Temporary variable*
                $d_{aux} \leftarrow minimum(d_{aux}, d_{temp})$
            end for
            $\mathbf{D}_{geo}(i, j) \leftarrow minimum(d_{aux}, \mathbf{D}_{SL}(j, i))$
            $\mathbf{D}_{geo}(j, i) \leftarrow \mathbf{D}_{geo}(i, j)$ ▷ This matrix is positive and symmetric
        end for
    end if
    return $\mathbf{D}_{geo}$
end procedure

$\mathbf{D}_{manifold} \leftarrow \mathbf{D}_{geo}$ ▷ Adjacency matrix of pairwise distances of Fisher manifold

mutually exclusive and their results can be complementary. Table 6.1 shows a quick comparison between two approaches.

TABLE 6.1 – Comparison table between network approach and cMDS embedding

FISHER INFORMATION NETWORK	
Advantages	**Disadvantages**
Works directly with adjacency matrix	Needs a neighbourhood parameter, σ_G
Can segment the network independently of its density	Densities can be masked by wrong σ_G
Similarities can used to estimate the class label	Cannot apply projective methods

FISHER MANIFOLD EMBEDDING	
Advantages	**Disadvantages**
Observations are described by coordinates instead of a network with edges and nodes	For Riemannian manifolds the embedding needs many dimensions to accurately reproduce the pairwise distances
Eigenvalue decomposition allows to know the relative weight of each dimension, and reduce its dimensionality at expenses of distance accuracy	Low dimensional embeddings imply a higher distance error
Manifold complexity measured by its required dimensionality	
Many algorithms available in Euclidean spaces, like PQC	

6.1.1.1 Example comparison 1: manifold of shopping baskets

In this subsection both clustering methodologies are applied to the shopping baskets manifold, this manifold is extracted from the retail data case study, Section 6.2, where it will be introduced with more details. For the purpose of this subsection, which is to illustrate the cluster differences, it is enough to know that the manifold is formed from shopping baskets and class labels related with customer loyalty. The loyalty labels are provided by the retailer company according to their internal indicators, but they are not strictly derived from shopping baskets. The loyalty label is sensitive information related to mainly to Recency and Frequency of purchases.

Figure 6.1 shows the embedded Fisher manifold with the original class labels defined by loyalty. The class labels are quite mixed, having two minority classes with low prevalence (ly3 and ly4).

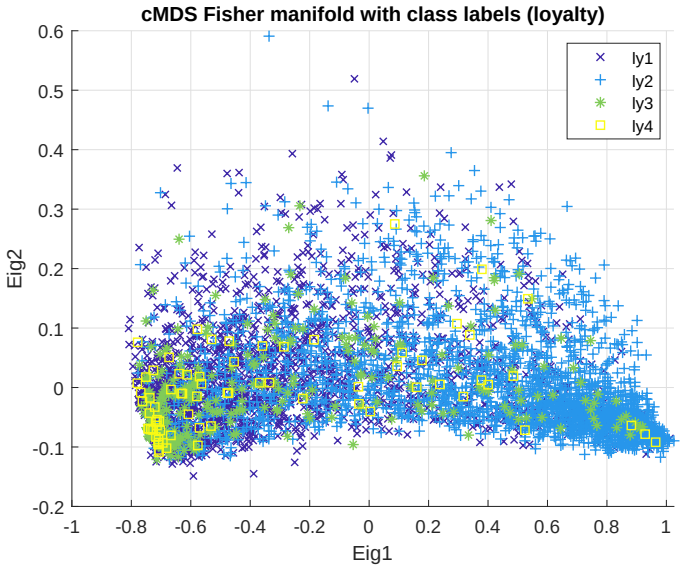


FIG. 6.1 – cMDS FIN manifold with loyalty class labels

Applying both clustering methods, two different cluster labels are obtained. Figure 6.1 shows the network communities with spectral clustering applied to the FI network. The length scale of the network has been computed with the heuristic method 3.2.3.2, obtaining a $\sigma_G = 0.1399$. The legend indicates the percentage of the majority class respect to each community. This community finding algorithm segments the manifold across its main eigenvector direction, creating communities of approximately the same size.

On the other hand, figure 6.3 shows the cluster labels of PQC applied to this embedded space. The legend indicates the percentage of the majority class respect to each cluster. The hyper-parameters are chosen following the ANLL score from figure 4.2.5.2. The clusters found by PQC have not similar sizes, there is a big cluster (cluster 1) that covers most of the middle part of the manifold, there are two other clusters located at the ends of the manifold (clusters 2 and 3) with half the population of the largest, these clusters have higher density and contain observations (customers) with more polarized loyalty behaviour. And there is a little cluster (cluster 4) with 7 observations that PQC has separated from the main region, it can be discarded as spurious.

Additionally, PQC provides details about the conditional probabilities associated with the cluster membership given the data, shown in figure 6.5, and also the conditional probabilities to belong to any cluster K, which can be used to detect outliers, this is shown in figure 6.6.

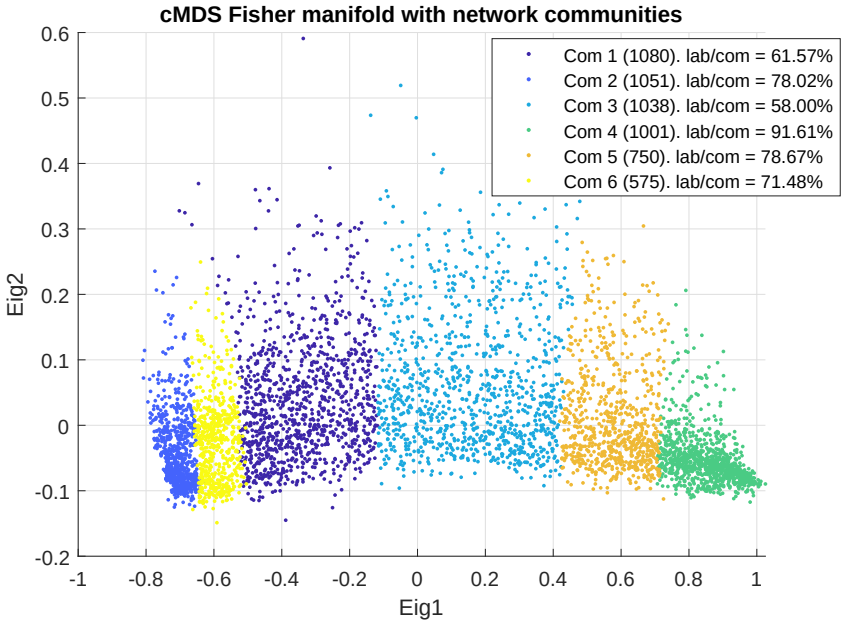


FIG. 6.2 – Loyalty FIN manifold with network communities

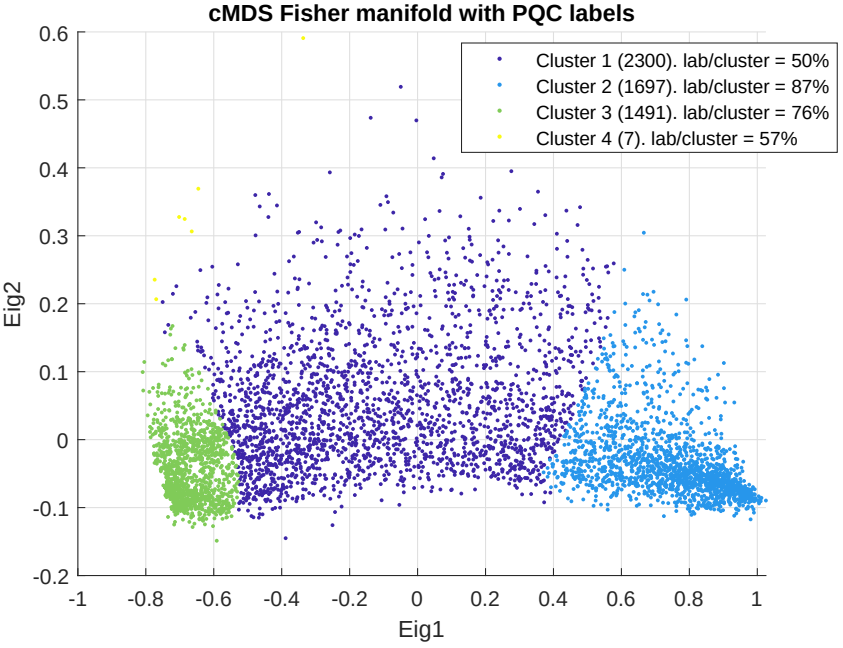


FIG. 6.3 – Loyalty FIN manifold with PQC clusters

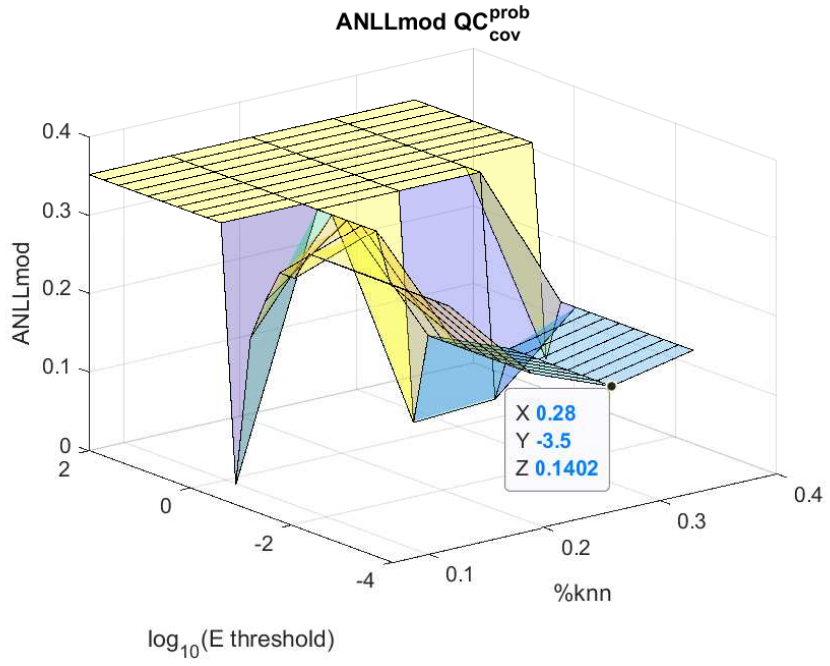


FIG. 6.4 – ANLL score for loyalty manifold

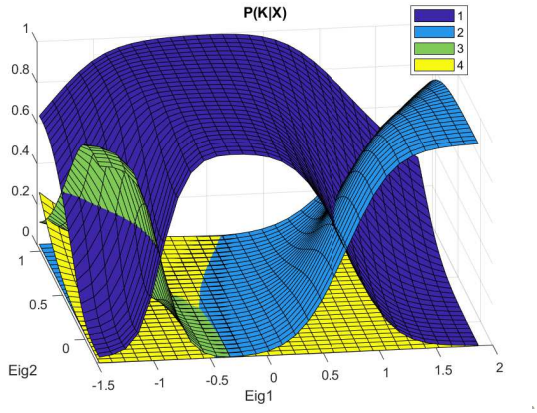


FIG. 6.5 – Loyalty PQC probability: $P(K|\mathbf{X})$

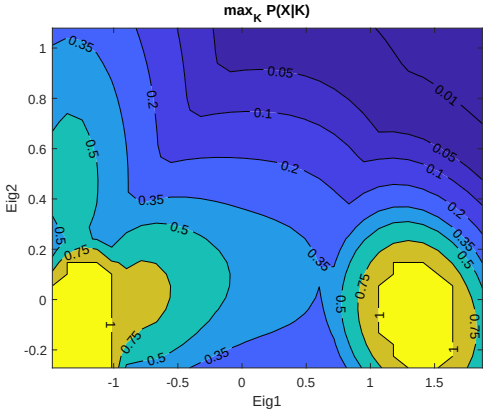


FIG. 6.6 – Loyalty PQC probability: $P(\mathbf{X}|K)$

Comparing the labels of both cluster methods, they are quite similar and consistent. The main difference is that the spectral clustering has segmented in more communities the manifold, but keeping the main cluster shapes. For instance, the PQC cluster 1 is almost equivalent to the communities 1 and 3, the PQC cluster 2 includes the communities 4 and 5, and the PQC cluster 3 is equivalent to the communities 2 and 6. With these blocks the percentage of the majority class respect to clusters are similar. Analysing the labels similarity quantitatively, the Cramer's V statistic [71] can be useful to measure the level of matching between PQC labels, communities and class labels, in eq. 6.1 is shown the level of agreement between them. Both methods have a similar score between them (0.7738) and both have a low score respect to the class labels (0.35), this is due to the two minority classes that have not been separated by any cluster.

$$
\begin{aligned}
C_V\left(class, clusters_{PQC}\right) &= 0.3414 \\
C_V\left(class, communities\right) &= 0.3590 \\
C_V\left(communities, clusters_{PQC}\right) &= 0.7738
\end{aligned}
\tag{6.1}
$$

6.1.1.2 Example comparison 2: manifold of music dataset

This subsection is another example comparison of both clustering methodologies applied to the music Fisher manifold, this manifold is extracted from the music data case study, Section 6.3, where it will be introduced with more details. For the purpose of this subsection, which is to illustrate the cluster differences, it is enough to know that the manifold is formed from music songs described by their spectral features where the class labels are the musical genre. The musical genre has been externally labelled by the artist, producer or the society in general, but not specifically according to their spectral features.

Figure 6.7 shows the embedded Fisher manifold with the genre labels. All genres have similar population. The manifold has triangular shape, the manifold corners are regions with only one predominant music genre, although the labels are quite mixed in the intermediate regions, which means that the songs are a fusion of styles.

sigma 0.41381

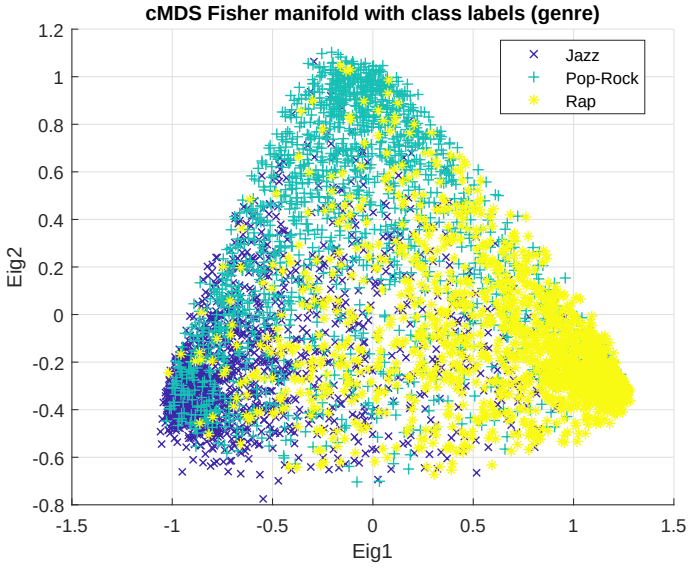


FIG. 6.7 – cMDS FIN manifold with loyalty class labels

Figure 6.8 shows the communities detected by the network. The length scale of the network is $\sigma_G = 0.4138$ and it has been computed with the heuristic method 3.2.3.2. The legend indicates the percentage of the majority class respect to each community.

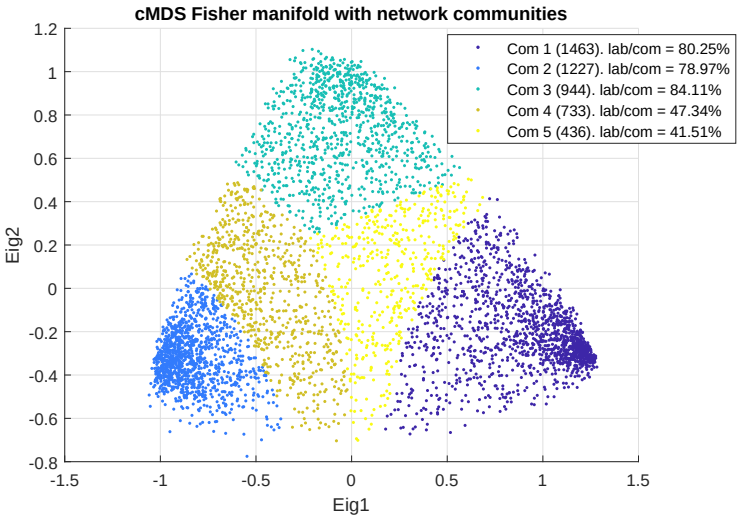


FIG. 6.8 – Music genre FIN manifold with network communities

Figure 6.9 shows the clusters detected by the PQC. The hyper-parameters have been selected according the same ANLL plot 6.46 depicted in Subsection 6.3.4.3. And the plot of probability of cluster membership, $P(\mathbf{X}|K)$, is also depicted in figure 6.47.

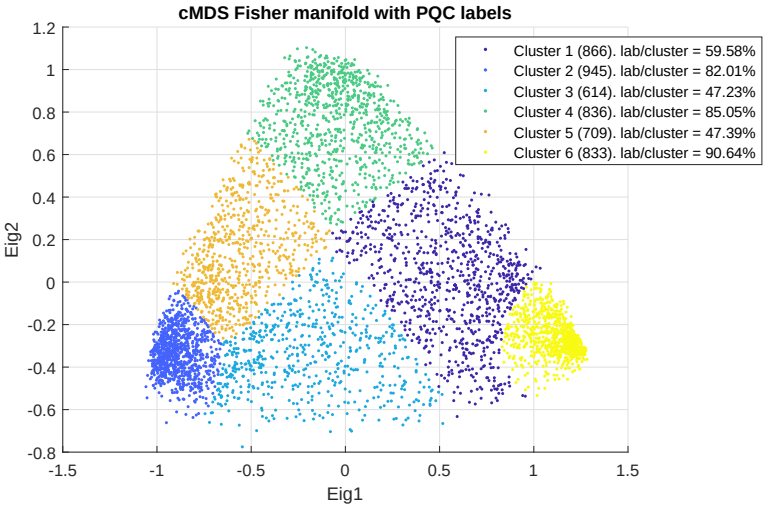


FIG. 6.9 – Music genre FIN manifold with PQC clusters

Comparing the labels of both clustering methods, one can observe the PQC captures better the high density regions than the spectral clustering with Newman's algorithm. This fact is reflected in the percentages of the majority classes respect to clusters, in the clusters associated with regions of dominant genres. Next table 6.2 shows the percentage comparison:

TABLE 6.2 – Percentage of majority class respect to the cluster

Class / cluster ratio	**Pop-rock**	**Jazz**	**Rap**
Communities	84.11%	78.94%	80.25%
Clusters	85.05%	82.01%	90.64%

Measuring the level of matching between the class labels and the clusters, similar results to the previous example have been obtained. Both methods have a 0.7703 score of agreement (being 1 the maximum) which is very similar to the example 1 with a 0.7738. However, in this example the agreement with the class labels is better (0.63) than the retail case (0.35).

$$\begin{aligned} C_V\left(class, clusters_{PQC}\right) &= 0.6333 \\ C_V\left(class, communities\right) &= 0.6351 \\ C_V\left(communities, clusters_{PQC}\right) &= 0.7703 \end{aligned} \tag{6.2}$$

Therefore, roughly it can be concluded that there is not a huge difference in the labels of both methods, although there are nuances in terms of density or segmentation that can be enhanced as a function of the manifold distribution.

6.2 Retail data case study

This section applies the FIN pipeline to the customer shopping baskets using the data provided by a large retail UK company. The main difference from the previous workflow implementations is that the communities obtained from the Fisher manifold will be used to build CI-maps to analyse customer behaviours based on the products associations defined by community shopping baskets.

6.2.1 Motivation of retail data

A substantial component of this thesis is the work developed in collaboration with a large UK retailer company. The objective was to find customer insights applying the proposed procedure in the context of customer longitudinal transactions data, segmenting communities as a function of class labels defined by the company. This kind of data presented several challenges: a) high dimensionality of products as features, b) noise

associated with a real scenario, c) software robustness needed to work in production, and d) scalability of the algorithm to work in a big data environment.

For a retail company the proposed pipeline is useful for the following reasons: Customer insights are crucial for optimizing the consumer experience. As part of this, customer segmentation is one of the most important topics in a retail business [130–132]. The total set of consumers can be segmented by analysing customer transactions, shopping baskets (products purchased), order history, and other factors. But depending on the classification criteria, the customer can be segmented in completely different ways, for instance by loyalty, by brand of products, or by Recency-Frequency-Monetary value (RFM model). Each criteria produces a different type of segmentation in shape and number.

One of the key questions lies in having the same customer dataset, generate completely different results depending on the class labels used. The FI metric modifies the dataset topology as a function of the classification criteria in order to enhance the label separation. The classification criteria can be understood as business questions that generate a different set of labels for the training data. These labels help to identify each cluster in a semi-supervised way, hence once the network has been built a new consumer can be allocated to the closest segment. If a different business question is applied, different labels produce a different metric and therefore a different network and segmentation. The aim is to obtain a homogeneous group of consumers segmented by a business question.

Other key questions arise once the consumer segment has been identified, for instance, determine if the identified products act as a driver of other sub-products. In other words, this implies that an association map which relates a product's dependencies can be constructed. Using the state-of-the-art techniques in structure finding algorithms based on conditional independence maps, Bayesian Networks can be built that relates products within each segment. These networks can be very useful in terms of product promotions, customer product switching (branded versus own-labelled, healthy versus not-so-healthy, or other combinations) and other marketing objectives.

The last key question is scalability, the volume of data involved in consumer transactions forces the segmentation techniques to be scalable. The nature of the data is complex; considering the sample size as the number of consumers, it can be around several millions in the case of large companies like the collaborating retailer in this project. Considering each product as a feature in the dataset, this produces a high dimensional dataset. Additionally, if each observation belongs to the consumer shopping basket, it makes the data very sparse.

From the perspective of real world applications, the whole framework of FI networks, communities and Bayesian networks was applied to real customer retail data. One of the novelties resides in the use of shopping basket data to extract customer insights using the full procedure. The fact of dealing with real data implies additional tasks such as data preprocessing, data cleaning, feature reduction, and also additional efforts to make results interpretable.

The case study of loyalty as a business question reveals that noisy real-world data combined with high dimensionality tends to favour linear classification models, where linear models are equally good as non-linear models. The embedded Fisher manifold presents a 1-dimensional cigar shape distribution, characteristic of linear models, where the customers are uniformly distributed between two extremes, loyalty and churn. Communities are segmented across this cigar shape, being in the extreme the communities that indicate a more characteristic shopping basket profile, driven by the total spending in certain classes of products. The customer spending in these products are more sensitive to the loyalty label. At the same time, analysing the CI-maps for each community, the same class of products are the central nodes in the map, acting like a hub where most of the other products are connected to them. From the marketing point of view, these products are interesting in terms of promotions or loyalty improvement.

6.2.2 Introduction

The aim of this study is to segment customers into communities with similar shopping basket behaviour. This is achieved by finding a statistically rigorous metric for the available attributes of the model, in this case spending values by product category, which is informed by an class label representing a given indicator of relevance to the business.

A similarity metric is inferred from the posterior probability map for the class label, in this case a measure of customer loyalty. Aggregated shopping basket data is mapped into a Riemannian space using the Fisher Information metric and a Fisher Information Network (FIN) is derived using geodesic pairwise distances converted into similarity values. The Fisher metric creates a data structure that reflects the business question indicated by the class labels, by bringing together shopping baskets with similar outcomes of the given class label and distancing shopping baskets either side of classification boundaries. Communities in this network define clusters of shopping baskets by similarity according to the metric induced by the class label. Each community is likely to have a characteristic shopping behaviour. This behaviour is characterised by a multivariate correlation model represented by the Conditional Independence Map, or graphical model

that reflects the statistical associations between product categories. This takes into account multiple levels of conditioning and therefore, removes spurious associations i.e. it controls for False Positives.

It was found that statistical models of loyalty map the data into an approximately linear space with gradual changes in the prevalence of loyalty and churn. This was found with separate models for distinct cohorts defined by an affluence label, so as to remove this attribute as a confounder for loyalty. In this context, churn can be understood intuitively as the opposite to loyalty, the churn rate usually measures the proportion of contractual customers who leave a supplier during a given period of time.

Communities were identified with prevalence of loyalty of 92% and churn of 78%, which is higher than the sensitivity and specificity of a binary classifier. The different communities are characterised by product category associations with similar basket profiles, with c10, c76 and c69 the class-products most sensitive in terms of the communities defined by the loyalty label.

6.2.3 Methodology

The methodology follows the algorithm described in the FIN stages of processing, the main differences lie in the pre-processing section with an extended analysis based on feature reduction and class label dependencies. For the community detection Newman's algorithm (spectral clustering) has been applied in preference to PQC. This is due to the Fisher manifold presenting a homogeneous distribution, which is a segmentation problem, so being suitable to spectral clustering algorithms, or also K-Means if it was applied directly to an embedded Fisher manifold. Once the communities are identified, then the CI-maps methodology is applied per communities.

The main pipeline is described as follows:

- Data pre-processing for feature reduction: 80/20 rule, features grouped hierarchically, $\mathbf{X} \rightarrow log(\mathbf{X}+1)$ feature transformation, feature selection based on CI-maps and data stratification.
- MLP performance analysis on different class labels.
- Fisher pipeline applied to data stratified by *loyalty* labels.
- cMDS on Fisher manifold
- Dominant first eigenvector allows projective methods and distribution analysis on this direction.

- Community detection with Newman's algorithm, and shopping basket profiles.
- CI-maps and customer behaviour analysis based on each *loyalty* community.

6.2.4 Data description

The dataset is an anonymised sample of 30K customers who have 3-months-worth of transactional data and have shopped at least five times.

The customer transactions are listed with the lowest product level identifier, called *tpnb*. The products act as features when the data is organized as shopping baskets. However, the company also provided the hierarchical groups where the products can be organized by higher category levels. The following list enumerates the hierarchical feature levels, with the number of features stated in brackets:

1. Commercial hierarchy division (10)
2. Commercial hierarchy group (34)
3. Commercial hierarchy department (100)
4. Commercial hierarchy class (318)
5. Commercial hierarchy subclass" (1317)
6. tpnb (10813)

Using higher category levels reduces the number of features, but also implies a loss in the customer transaction detail given many products are aggregated into lower groups. The objective is to find a trade-off between transaction detail (useful for the MLP performance), and few features (useful for reducing the data sparsity and the runtime).

In addition, a set of different class labels were provided by the customer organisation and were used as an class labels. For instance, the loyalty label is sensitive information related to mainly to Recency and Frequency of purchases. Analogously, the other labels measure different customer indicators.

These labels can be used to train the MLP, hence altering the structure of the Fisher manifold. There are five class labels, each with multiple attributes stated in brackets:

1. *Loyalty (4)*
2. *On-line loyalty (9)*

3. *Life-style (3)*

4. *Life-stage (5)*

5. *Affluence (4)*

All labels are sorted in increasing order, in other words, *Loyalty 1* < *Loyalty 2* < ... < *Loyalty 4*; being *Loyalty 4* the highest grade of loyalty.

Customer records of both datasets, transactions and customer labels, are linked by its household number.

6.2.5 Data preprocessing

The original data is transformed with the purpose of creating shopping baskets per customer, where each row represents a customer and each column represents a product. The matrix contains the total amount spent per product and per customer. There are five possible aggregations of shopping basket depending on the hierarchy level chosen.

In order to reduce the number of different products, the total amount spent in each product is used to sort and filter the products that represent 80% of the total spend. With the expectation that the dataset follows the 80/20 rule, where 80% of the spend represents 20% of the products bought.

The data is also cleaned of missing values in the main class labels, *on-line loyalty* label has been excluded due to the high amount of missing values (84%).

For the Fisher Information metric it is necessary to obtain a probability map of the shopping basket space, for this purpose a Multilayer Perceptron (MLP) was used. The shopping basket data is sparse, containing many products as features with zero spend. To improve the MLP classification, each feature (column) is transformed with $\mathbf{X} \rightarrow log(\mathbf{X} + 1))$ to widen the shopping basket distribution, $\mathbf{X}$ represents the spending and always is equal or greater than zero.

For the best trade-off in the MLP training over the five hierarchy levels, product category *commercial hierarchy class* was chosen which after the 80/20 filtering contained 130 products. This level provides a good representation of the shopping baskets with less noise in the data than lower level categories, and is less biased or mixed than the higher categories. After the pre-processing there were 23K customers (rows) and 130 class-products (columns), and 4 class labels.

6.2.6 Feature selection with CI-maps and affluence stratification

The final step is to build a Bayesian Network from Conditional Independence map (CI-map) from the Fisher manifold communities. The CI-maps performance depends on the ratio of degrees of freedom (DoF) and the sample size. After the feature reduction using the 80/20 rule, there are still 130 features (class-products) which is too many DoF to properly perform the conditional independence tests given the stratified data subset.

Although the MLP can successfully train with many dimensions, the belief is that many of the products are not adding useful information to the classifier model. Hence, it would be useful to apply an additional dimensionality reduction technique for the model, helping at the same time the posterior CI-map.

One option to reduce the dimensionality is at the same time as constructing the CI-map is to apply the feature selection process as described previously in chapter 5.2.4. Recalling that the technique consists of building a CI-map with the class label and the products with the purpose of identifying which products have a dependency (edges) with the targeted label. As pre-processing is required to combine a dataset with the target label (*loyalty*) and the shopping baskets with the product spending discretized into quantiles (2, 3 and 5 quantiles were tested, with 3 quantiles as the most balanced). The criteria to select those products that are associated with the class label is to choose the set with first and second order of connection with the class label.

As an example, in the Subsection 6.2.6.1 there is the figure 6.10 that shows the CI-map of *loyalty* vs 130 class-products. Although the product identifiers are not very legible you can observe the structure and the main product relationships of the whole shopping basket. Applying the CI-maps for feature selection the products reduce from 130 to roughly 30 class-products.

By definition, *affluence* is sensitive to the total spending in the shopping baskets, and intuitively this is the case for *loyalty* too. In Subsection 6.2.6.2 independence tests between the class labels has been undertaken, showing that most of the class labels are dependent, especially *affluence-loyalty* shown in 6.3. To avoid affluence having impact in the *loyalty* analysis the dataset has been stratified by the affluence labels, in such a way that four subsets have been created, one per affluence label. The objective is to isolate the *loyalty* behaviour from affluence. By stratifying the data, the sample size is reduced to roughly 7.5K observations with the result that the runtime is significantly improved.

This analysis of shopping baskets is therefore stratified by affluence, although at the end of the chapter there is a Section 6.2.9 where the data has not been stratified, and provides similar results to the stratified cases.

6.2.6.1 Feature selection of products with CI-maps

The figure of this Section 6.10 shows the CI-map applied to the *loyalty* label as target variable, where the 130 class products conform a big CI-map. Although the product nodes are not legible due to the font size in the big graph, one may observe the main class-products associations, where there are some central nodes that act as a hub, setting a dependence with many products. The loyalty label is highlighted in red circle and the first order nodes are in green circles. The 33 second order nodes are not highlighted because the figure would be very messy.

6.2.6.2 Independence tests of class labels

Another point to analyse before applying the Fisher pipeline is the independence between the class labels. Next tables, from 6.3 to 6.8, show independence tests of class labels by pairs. In almost all independence tests the null hypothesis, which assumes independence, can be clearly rejected with a $p-value < 0.05$, the unique exception is *Loyalty* - *Lifestyle*, where the $p - value = 0.0525$ is close to the decision value. But approximately all the class labels can be considered dependent.

TABLE 6.3 – Independence test: Loyalty - Affluence

Loyalty \Affluence	**lst2**	**lst3**	**lst4**	**lst1**	χ^2
ly4	0.5%	13.7%	14.1%	1.9%	76.42
ly2	0.3%	8.8%	10.4%	1.8%	***p-value***
ly1	0.4%	10.9%	12.3%	1.6%	8.27E-13
ly3	0.3%	11.2%	10.4%	1.3%	

TABLE 6.4 – Independence test: Loyalty - Lifestyle

Loyalty \Life-style	**lst2**	**lst3**	**lst4**	χ^2
ly4	0.4%	0.7%	0.4%	12.45
ly2	14.5%	19.5%	10.8%	***p-value***
ly1	15.9%	20.0%	11.3%	0.0525
ly3	2.3%	2.7%	1.6%	

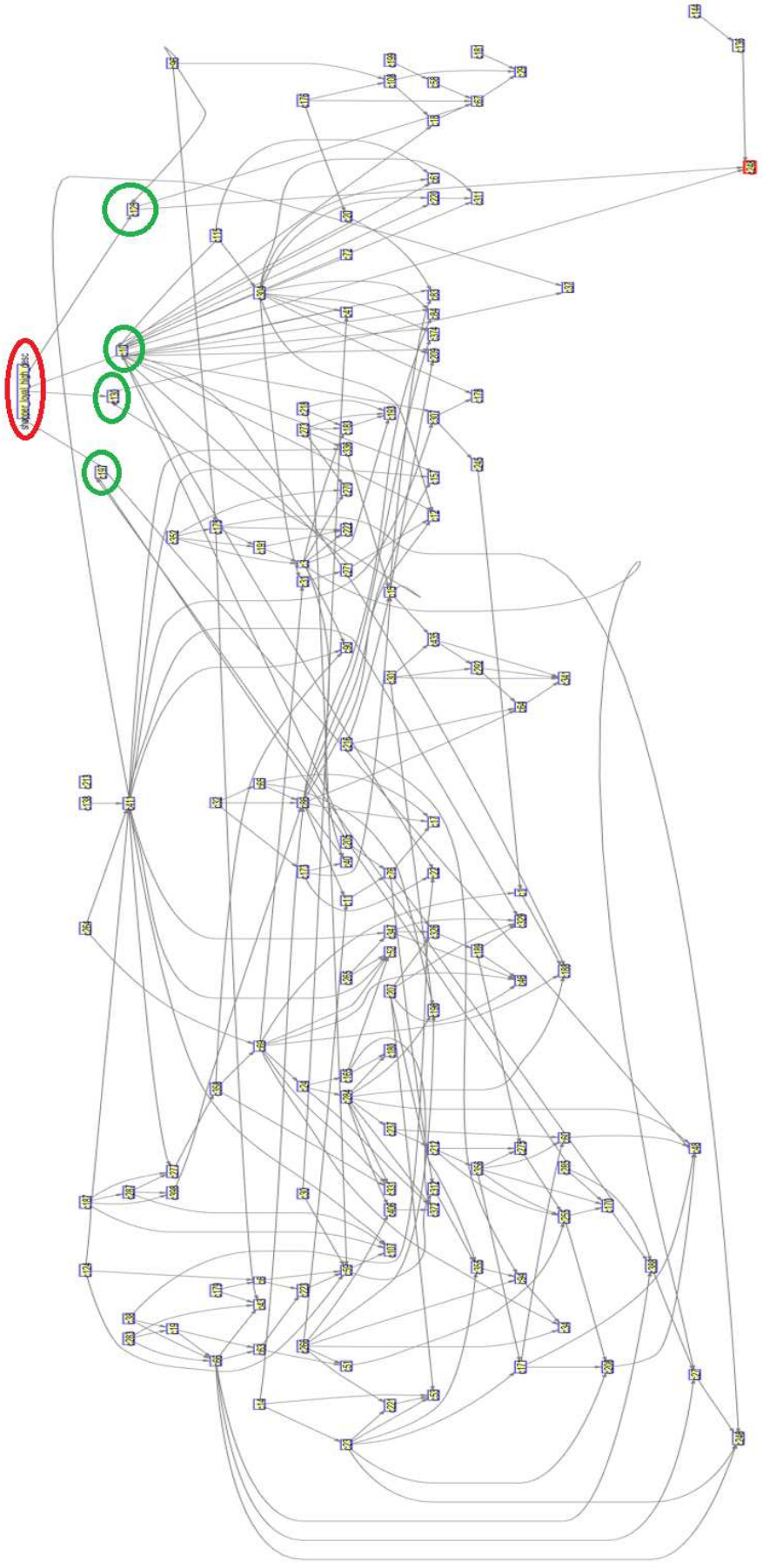


FIG. 6.10 – Loyalty CI-map of 130 class products

TABLE 6.5 – Independence test: Loyalty - Life-stage

Loyalty \Lifestage	**lst3**	**lst5**	**lst2**	**lst1**	**lst4**	χ^2
ly4	0.4%	0.2%	0.4%	0.3%	0.2%	917.80
ly2	4.5%	9.4%	12.0%	10.2%	8.7%	***p-value***
ly1	10.7%	8.2%	12.1%	10.3%	5.8%	8.63e-189
ly3	1.9%	1.2%	1.6%	1.1%	0.7%	

TABLE 6.6 – Independence test: Affluence - Lifestyle

Affluence \Lifestyle	**lst3**	**lst5**	**lst2**	χ^2
ly4	9.3%	13.0%	7.9%	291.17
ly2	8.1%	9.5%	3.6%	***p-value***
ly1	8.6%	11.1%	5.6%	6.35e-60
ly3	7.1%	9.1%	7.0%	

TABLE 6.7 – Independence test: Affluence - Lifestage

Affluence \Lifestage	**lst3**	**lst5**	**lst2**	**lst1**	**lst4**	χ^2
ly4	5.3%	5.0%	8.2%	7.2%	4.5%	868.03
ly2	4.3%	5.7%	4.8%	2.8%	3.6%	***p-value***
ly1	4.9%	5.5%	6.3%	4.8%	3.7%	4.18e-178
ly3	3.1%	2.9%	6.7%	7.0%	3.6%	

TABLE 6.8 – Independence test: Lifestyle - Lifestage

Lifestyle \Lifestage	**lst3**	**lst5**	**lst2**	**lst1**	**lst4**	χ^2
ly4	4.3%	4.9%	8.5%	11.3%	4.2%	1366.3
ly2	7.7%	10.3%	10.3%	6.7%	7.7%	***p-value***
ly1	5.6%	3.9%	7.3%	3.9%	3.5%	1.11e-289

6.2.7 MLP performance with different labels

The MLP performance has been shown to be the key to creating structured FINs where the communities truly represent a good segmentation of the class labels. In this section, the train and test accuracy of the MLP using the main class labels are presented, with two pre-processing options: z-scoring and row normalization.

The MLP model has been trained (70%) and validated (15%) with the three different sets of class labels and two kind of preprocessing, the selection criterion has been the accuracy of the test set (15%), which is the best score to check if the Fisher metric derived from the MLP probabilities can map the input space with the class labels. Inspecting MLP accuracy in table 6.9 the following conclusions can be made:

- Only *loyalty* provides accuracy ($> 70\%$). With the CI-map feature selection, the MLP test accuracy is the same for both, 130 or 37 features, indicating there is considerable redundancy in the features. See figures 6.11 and 6.12.

TABLE 6.9 – MLP performance vs class labels

Label	Features	Z-score	Row norm.	Train Acc.	Test Acc.
Loyalty	130	No	No	72.8	**71.4**
			Yes	68.3	66.5
		Yes	No	72.8	71.2
			Yes	65.8	65.8
	37	No	No	72.0	**72.3**
			Yes	65.5	64.4
		Yes	No	71.7	71.6
			Yes	51.2	51.9
Affluence	130	No	No	34.4	33.0
			Yes	30.1	31.0
		Yes	No	33.7	**33.4**
			Yes	32.6	32.2
	32	No	No	31.6	31.6
			Yes	30.7	31.9
		Yes	No	31.4	31.5
			Yes	23.8	23.5
Life-style	130	No	No	54.5	53.0
			Yes	53.8	53.6
		Yes	No	55.6	**53.6**
			Yes	54.6	52.8
	35	No	No	51.0	49.2
			Yes	49.7	49.2
		Yes	No	50.8	50.1
			Yes	33.1	33.4

- Customer normalization decreases the *loyalty* MLP performance, meaning that the total amount spent per customer is important in discriminating this label. However, this is not the case for the other labels.
- In general, Z-scores the data do not influence the result, at least in case of *loyalty*.
- *Affluence* and *life-style* have poor MLP performance, basically because there is no enough information in the shopping baskets to discriminate these class labels. Additional pre-processing has no effect on the performance.

Due to the poor MLP performance in the other class labels, only the *loyalty* label will be used in the Fisher procedure.

To illustrate the CI-map feature selection has worked well, figure 6.11 shows the *loyalty* MLP performance of a model with 130 features, and figure 6.12 the performance of a model with only 37 features. Both models have approximately similar results.

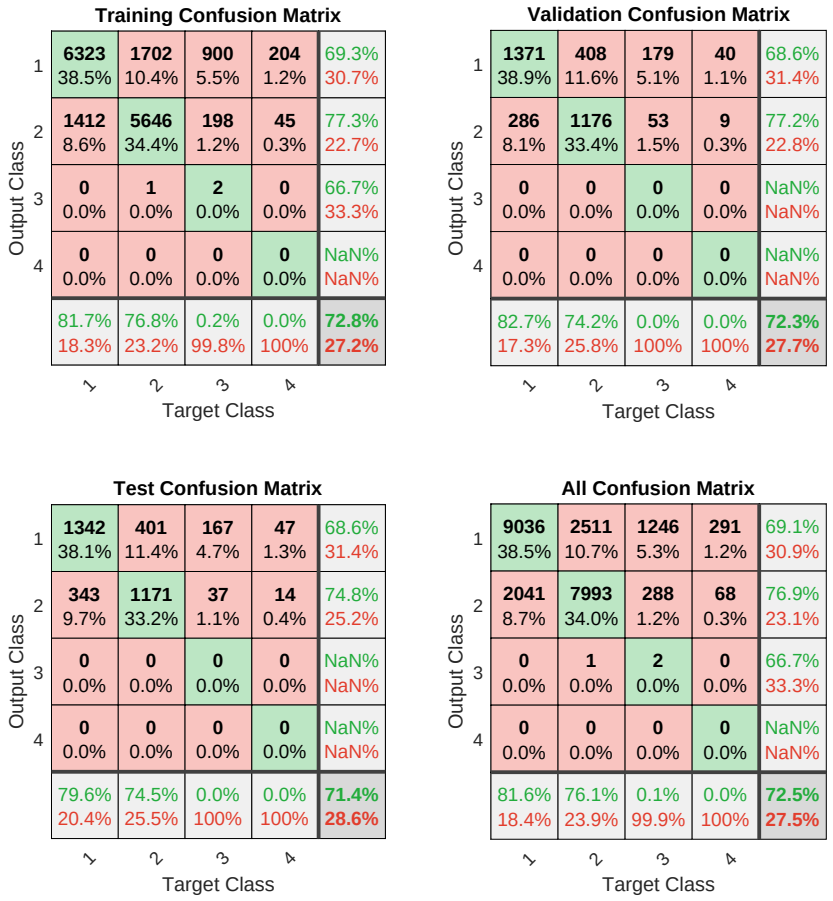


FIG. 6.11 – Loyalty MLP performance with 130 features

6.2.8 Loyalty FIN on stratified data

At this point the MLP has been trained, the FIN calculated, the Fisher Information metric has been defined and the pairwise distances have been optimized with a shortest path algorithm, using the Spark implementation for the pairwise distances.

This section shows the structure of the Fisher manifold with cMDS, the spectral clustering communities and the histograms obtained from the main eigenvector projection, if the eigenvector explains more than 80% of the variance. As an example, the subset of *affluence 1* has been chosen to display the Fisher manifold structure, noting that the other *affluence* subsets have similar manifolds.

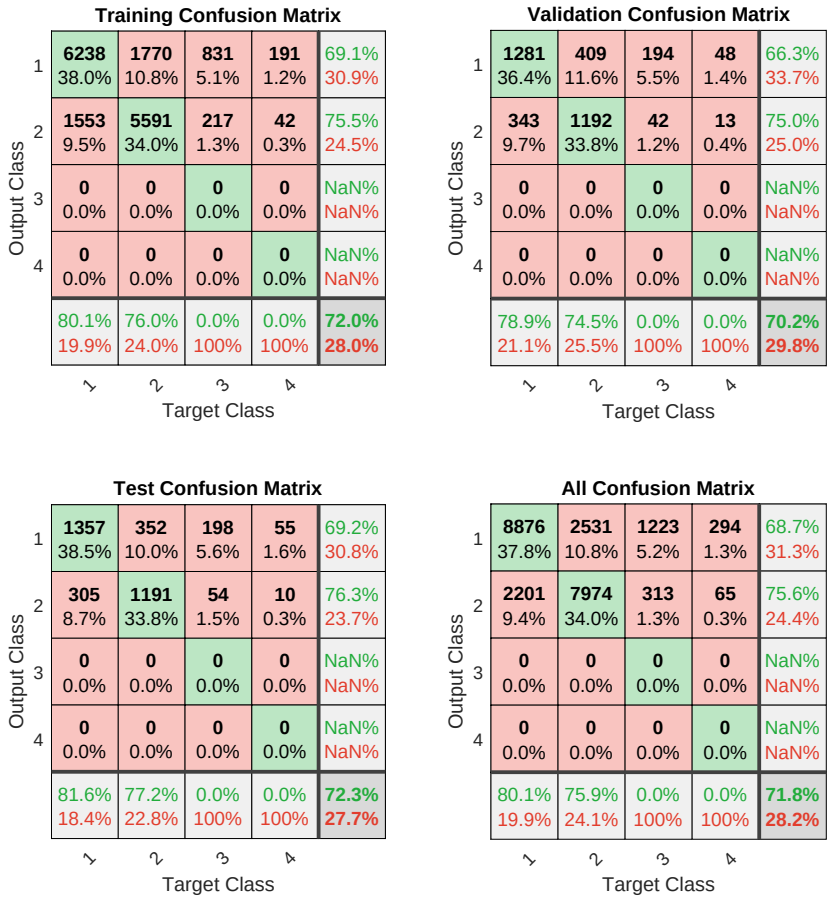


FIG. 6.12 – Loyalty MLP performance with 37 features

6.2.8.1 Structure of loyalty Fisher manifold with cMDS

The first step is to consider the MLP predictions for the *loyalty* labels, as the information captured by the MLP will determine the FIN structure. Figure 6.13 shows that the MLP only predicts two of the four class labels. This is mainly for two reasons: the low prevalence of labels *Ly3* and *Ly4*, and the noise in the data where customers with different *loyalty* labels are significantly mixed.

The main consequence for the FIN having a MLP with two labels is that it will behave similarly to a FIN based on a binary classifier, for example the one studied in Section 3.4.1. This means that the Fisher manifold structure will probably lie in one dimension.

Embedding the Fisher manifold with cMDS, figure 6.14, the *loyalty* manifold lies mainly in one dimension as expected, where in one extreme label *Ly1* dominates and in the

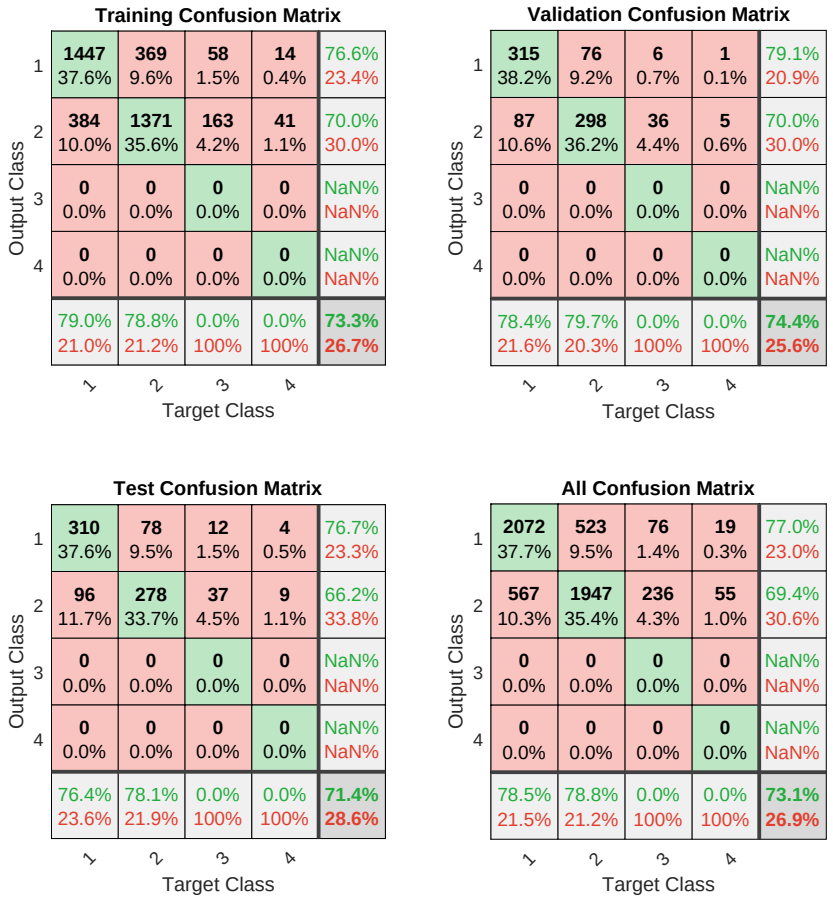


FIG. 6.13 – Loyalty MLP performance with data stratified by Affluence 1

other *Ly2*. The cMDS also shows the abundance of noise in the intermediate regions of the manifold, where there is significant mixing of labels.

Figure 6.15 depicts the eigenvectors of the *loyalty* FIN, where one may observe the 1-dimensional structure. On the other hand, representing the Euclidean version of the pairwise distances, you can see in figure 6.16 how the structure is more spherical due to the sparsity of the shopping baskets. Although in Euclidean space the *loyalty* labels are more mixed than in the FIN manifold, their first eigenvectors approximately discriminate *loyalty* labels *Ly1* and *Ly2*.

6.2.8.2 Loyalty probability histograms per affluence stratification

Given that the *loyalty* Fisher manifold has a 1-dimensional structure, projective methods can be applied, like histograms based on data projections on the first eigenvector. In

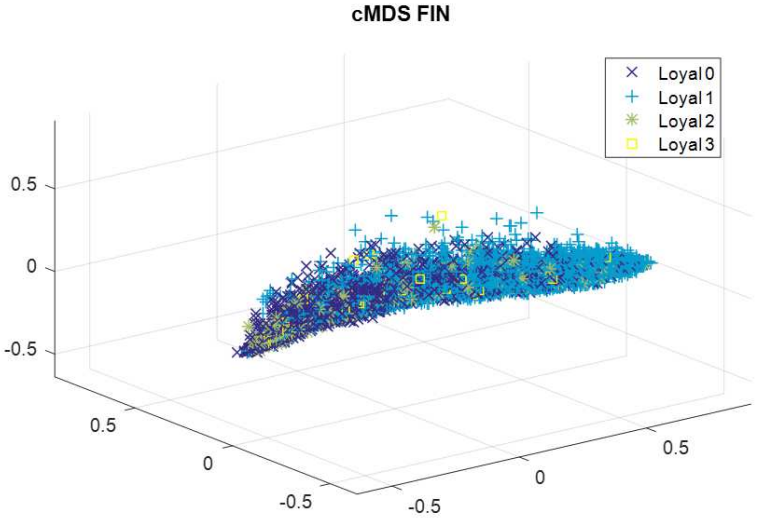


FIG. 6.14 – cMDS loyalty FIN manifold on Aff.1

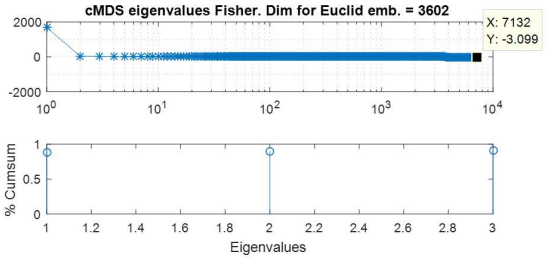


FIG. 6.15 – cMDS loyalty FIN eigenvalues on Aff.1

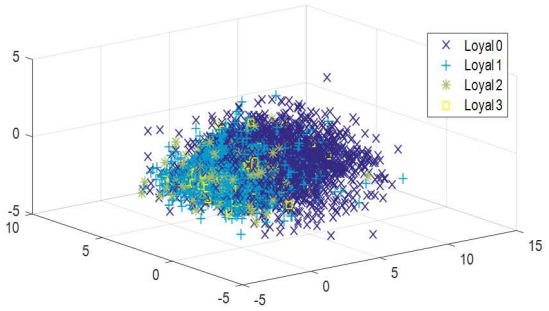


FIG. 6.16 – Loyalty Euc. dist. cMDS on Aff.1

this case, the interest is in the impact of the data stratification by affluence, specifically checking if the *loyalty* distributions change when different affluence data is considered.

Next figures, from 6.17 to 6.20, show the *loyalty* histograms applied to different affluence stratifications. Although the colour code associated with each label varies per figure, looking at the *Ly1* and *Ly2* labels, one may appreciate that the *loyalty* distributions are very similar independently of the affluence stratification.

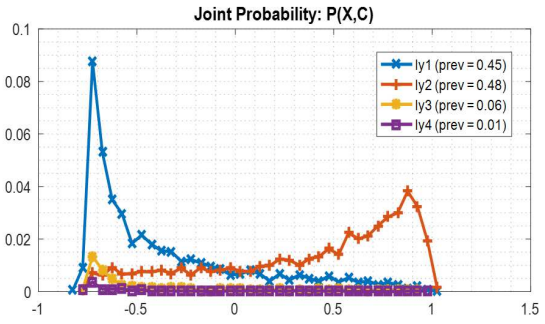


FIG. 6.17 – Loyalty histogram on Aff.1

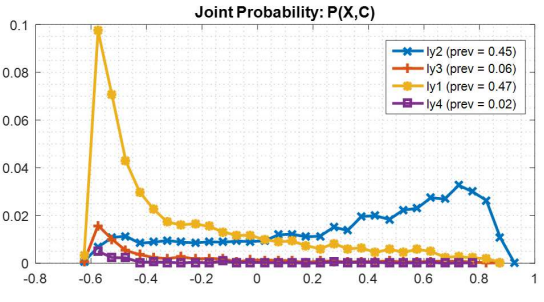


FIG. 6.18 – Loyalty histogram on Aff.2

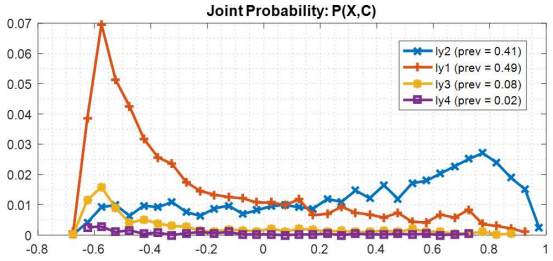


FIG. 6.19 – Loyalty histogram on Aff.3

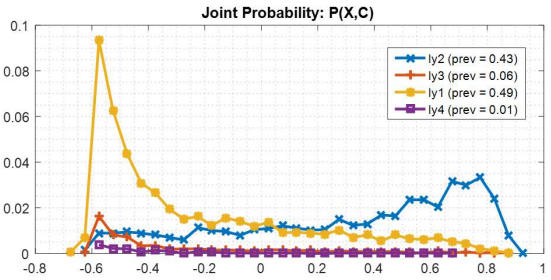


FIG. 6.20 – Loyalty histogram on Aff.4

All figures show a bimodal distribution, which relate to *Ly1* at the left of Fisher manifold, and *Ly2* to the right. Both distributions present a long tail in the intermediate regions, overlapping towards the centre. The higher *loyalty* labels with lower prevalence have no clear allocation in the manifold because they are not modelled by the MLP, by similarity *Ly3* is generally in the same location as the peak of the *Ly1*, there is no clear reason why the observations of *Ly3* should be similar to *Ly1* in the Fisher manifold.

The manifold structure is typical of binary distributions, where the Fisher metric pushes the observations that clearly belong to a specific label to one extreme, and the observations where the characteristics are less certain are placed in the intermediate regions, in such a way that the distribution represents a transition from one label to another. This 1-dimensional structure also reflects the fact that is MLP discriminating linear behaviour, which is an indication that a linear model such as binary logistic regression would be a suitable classifier.

6.2.8.3 Probability histograms with other labels

Using the Fisher manifold created using the *loyalty* labels, other labels distributions can be analysed in order to check if they are correlated with *loyalty*.

Figures 6.21 and 6.22 show the *Life-style* and *Life-stage* distributions respectively. Inspecting the figures it is apparent they are independent of the *loyalty* manifold, i.e. all categories have approximately the same distribution independently of the label. Therefore, it can be concluded the *loyalty* manifold does not provide any additional discrimination of *Life-style* or *Life-stage*.

6.2.8.4 Fisher manifold with combined labels

Another option for additional insights of the shopping baskets is to create combined class labels to create new Fisher manifolds. For instance, in table 6.9 the MLP accuracy of the *life-style* labels is very low, by combining *loyalty* and *lifestyle* labels a new class

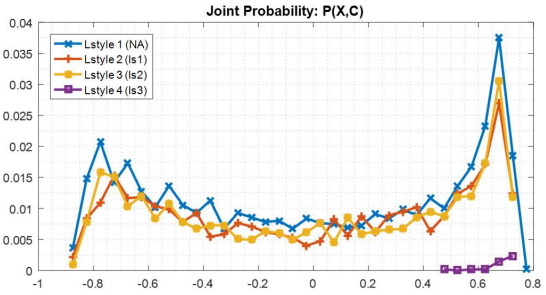


FIG. 6.21 – Life-style histogram on Aff.1 loyalty manifold

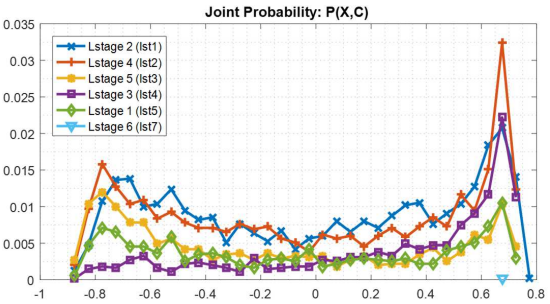


FIG. 6.22 – Life-stage histogram on Aff.1 loyalty manifold

label is created with better MLP accuracy. One way to reduce the number of categories is to use only those categories with highest prevalence, for example *[ly1, ly2]* of *loyalty* combined with *[ls1, ls2, ls3]* of *life-style*.

Figure 6.23 depicts the *loyalty-lifestyle* Fisher manifold. It has a slightly wider structure with a higher weight on the 2nd eigenvalue, but the distributions seem to be dominated by the *loyalty* labels. Figure 6.24 shows the probability distributions of the combined labels where one may observe the categories distributions are indeed driven by the *loyalty* labels, with the *lifestyle* distributions following the *loyalty* distributions. Therefore, it can be concluded that the shopping baskets do not contain information about *lifestyle*.

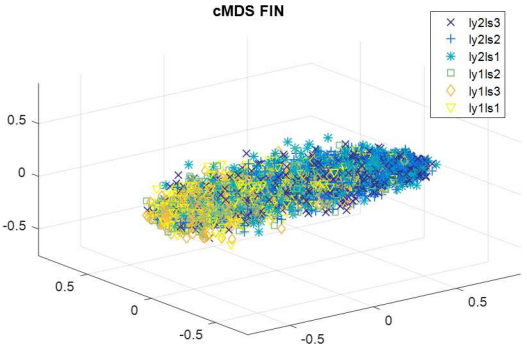


FIG. 6.23 – Fisher cMDS with combined label: Loyalty & Lifestyle

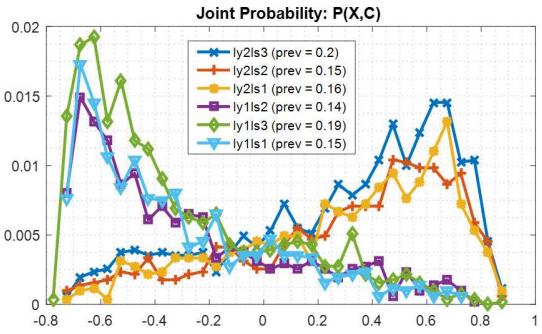


FIG. 6.24 – Loyalty & Lifestyle histogram on Aff.1

6.2.9 FINs without data stratification

It was shown in Subsection 6.2.8.2 that affluence does not affect the *loyalty* distributions. Other option is to build a Fisher manifold without the affluence stratification, and comparing for instance the affluence Fisher manifold with respect to the *loyalty* manifold. To reduce runtime a random sample of the original data was used for this analysis.

6.2.9.1 Affluence FIN

The main problem of the affluence label is the low performance of the MLP with the shopping basket data. Therefore, it is of interest to build the *affluence* manifold without taking into account the MLP accuracy. In figure 6.25 the cMDS structure is shown for this approach, although it presents a slight cigar shape, the *affluence* labels seem to be randomly distributed across the manifold.

Figure 6.26 shows the histogram based on the first eigenvector projection, where there is no correlation with the *affluence* labels. Again, similar conclusions to the *lifestyle* labels are obtained, the shopping baskets do not contain enough information to discriminate the *affluence* labels, and therefore the Fisher manifold is not effective when there is poor MLP performance.

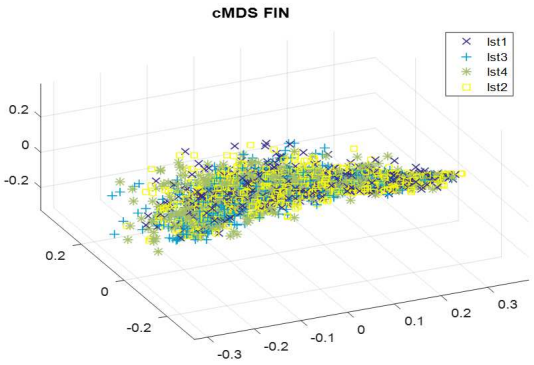


FIG. 6.25 – Affluence FIN cMDS

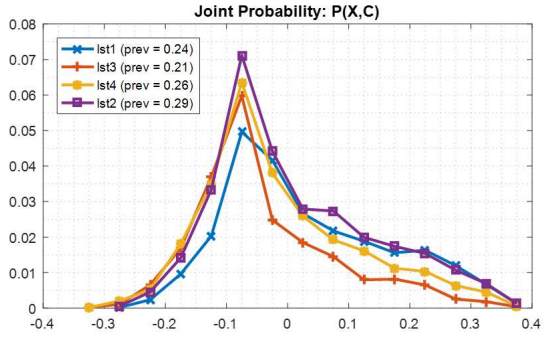


FIG. 6.26 – Affluence FIN histogram

6.2.9.2 Loyalty FIN

Applying the *loyalty* Fisher manifold to the same sample with no stratification, gives figure 6.27 where the *loyalty* distributions closely match the case with data stratification 6.2.8.2, and the *affluence* distributions in figure 6.28 continue to be independent of the *loyalty* manifold.

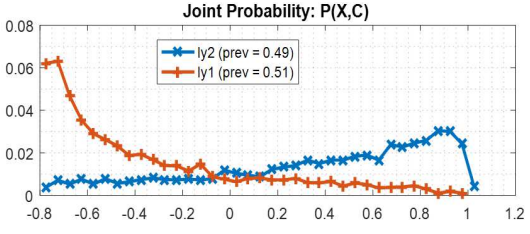


FIG. 6.27 – Loyalty FIN cMDS with no Aff. stratification

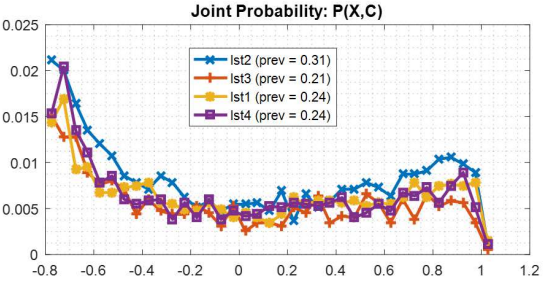


FIG. 6.28 – Loyalty FIN no Aff. strat. histogram

6.2.10 Community analysis based on loyalty FIN

In previous sections, the shopping baskets have been shown to only discriminate the loyalty labels. In this section, the shopping profiles created by the communities based on loyalty Fisher manifold are investigated. For example, the manifold created with the *loyalty* labels are applied to *affluence* 1, and the prevalence of loyalty within that category investigated.

Figure 6.29 depicts the cMDS Fisher manifold with the communities obtained with the Newman's algorithm. The cigar shape of the manifold structure transforms the clustering task into a segmentation task, where the communities are segments of the cigar shape. The extreme communities have higher ratios of the respective loyalty labels, *Ly1* on the left and *Ly2* on the right.

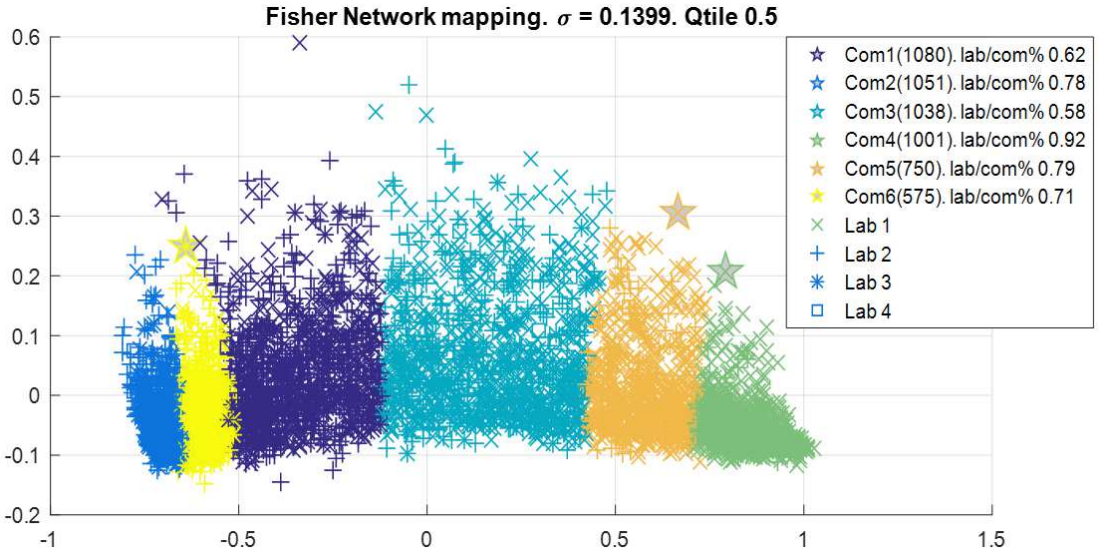


FIG. 6.29 – Loyalty communities on Aff.1

Using the communities to create shopping basket profiles can help to identify products that are more sensitive to the *loyalty* labels. The community profiles of all class-products are shown in the Subsection 6.2.10.1. Recalling the profiles are computed by the standardization of the mean community spending against the whole population, see Section 3.2.5.

Table 6.10 depicts the most relevant products in terms of absolute spending difference with respect to the whole population. The first column L, indicates the predominant *loyalty* label in the community, the second column C is the community identification, the third column R indicates the ratio of the maximum label membership within its community. The other columns are the standardized spending of the class-products, where the products with an absolute average spending greater than 0.5 have been colour

coded, orange for less spend compared to the whole population and yellow for more spend.

Looking at the product mean spend, there is a clear correlation with the *loyalty* and customer spending, the communities on the left of the manifold (2 and 6) are the least loyal and with less spend, while the most loyal are the communities 5 and 4. The intermediate communities 1 and 3 present a mix of loyalty behaviours with an average spend close to the whole population.

TABLE 6.10 – Community profiles of products most sensitive to FIN loyalty

L	*C*	*R*	*c10*	*c76*	*c347*	*c69*	*c115*
1	*2*	*0.78*	-0.93	-0.84	-0.72	-0.89	-0.75
1	*6*	*0.71*	-0.69	-0.56	-0.57	-0.60	-0.48
1	*1*	*0.62*	-0.33	-0.27	-0.34	-0.32	-0.32
2	*3*	*0.58*	0.10	0.10	0.06	0.09	0.06
2	*5*	*0.79*	0.57	0.52	0.45	0.58	0.42
2	*4*	*0.92*	1.20	1.00	1.05	1.09	1.03

6.2.10.1 Community profiles with all products

This section shows the full list all class-products per community profiles. Those products with absolute spending values greater than 0.5 are highlighted in colour. As mentioned in 6.2.10, the first column L, indicates the predominant *loyalty* label in the community, the second column C is the community identification, the third column R indicates the ratio of the maximum label membership within its community. The other columns are the standardized spending of the class-products. The orange colour is for the less spending than the whole population and the yellow colour for the greater spending than whole population.

Next table shows similar profiles but in this case for communities formed by the external *loyalty* labels.

6.2.10.2 Bayesian networks per loyalty communities

The following BNs are built from CI-maps based on the communities extracted from the loyalty Fisher manifold. The products associations that conform the shopping baskets are driven by the customer behaviour characterized by a loyalty similarity in the Fisher manifold. They are very important for extracting insights about the shopping basket structure and community behaviour. The BNs show that the class-products that act as hubs or central nodes in the graph, are the same class-products as in table 6.10, that present highest spending variations.

TABLE 6.11 – Community profiles with all products

L	*C*	R	c181	c197	c205	c265	c125	c10	c29	c285	c170
1	*2*	*0.78*	-0.13	-0.23	-0.30	-0.45	-0.43	-0.93	-0.51	-0.14	-0.25
1	*6*	*0.71*	-0.11	-0.13	-0.25	-0.31	-0.29	-0.69	-0.39	-0.08	-0.07
1	*1*	*0.62*	-0.04	-0.07	-0.11	-0.18	-0.15	-0.33	-0.26	0.03	-0.02
2	*3*	*0.58*	0.02	0.09	0.07	0.03	-0.02	0.10	-0.02	0.00	0.04
2	*5*	*0.79*	0.05	0.17	0.16	0.25	0.26	0.57	0.27	0.09	0.18
2	*4*	*0.92*	0.18	0.16	0.39	0.62	0.60	1.20	0.86	0.10	0.15

L	*C*	R	c48	c76	c165	c52	c347	c69	c248	c96	c108
1	*2*	*0.78*	-0.23	-0.84	-0.34	-0.30	-0.72	-0.89	-0.22	-0.26	-0.63
1	*6*	*0.71*	-0.20	-0.56	-0.26	-0.27	-0.57	-0.60	-0.08	-0.17	-0.45
1	*1*	*0.62*	-0.06	-0.27	-0.17	-0.20	-0.34	-0.32	-0.04	-0.07	-0.28
2	*3*	*0.58*	0.07	0.10	0.04	0.02	0.06	0.09	0.04	0.04	0.02
2	*5*	*0.79*	0.19	0.52	0.18	0.21	0.45	0.58	0.07	0.16	0.36
2	*4*	*0.92*	0.21	1.00	0.51	0.50	1.05	1.09	0.22	0.29	0.94

L	*C*	R	c213	c41	c72	c58	c176	c12	c83	c22	c271
1	*2*	*0.78*	-0.30	-0.32	-0.34	-0.42	-0.56	-0.60	-0.64	-0.54	-0.56
1	*6*	*0.71*	-0.10	-0.21	-0.22	-0.27	-0.37	-0.40	-0.42	-0.38	-0.45
1	*1*	*0.62*	-0.06	-0.10	-0.13	-0.19	-0.19	-0.16	-0.26	-0.22	-0.22
2	*3*	*0.58*	0.02	-0.01	-0.01	0.05	-0.08	0.06	0.02	0.00	0.02
2	*5*	*0.79*	0.19	0.12	0.21	0.17	0.24	0.34	0.32	0.28	0.38
2	*4*	*0.92*	0.29	0.47	0.48	0.63	0.91	0.70	0.93	0.81	0.79

L	*C*	R	c61	c220	c374	c188	c115	c264	c287	c311
1	*2*	*0.78*	-0.64	-0.63	-0.61	-0.61	-0.75	-0.71	-0.70	-0.70
1	*6*	*0.71*	-0.49	-0.50	-0.45	-0.52	-0.48	-0.53	-0.43	-0.58
1	*1*	*0.62*	-0.24	-0.27	-0.27	-0.29	-0.32	-0.36	-0.27	-0.35
2	*3*	*0.58*	0.05	0.02	0.00	0.05	0.06	0.06	0.06	-0.03
2	*5*	*0.79*	0.34	0.34	0.32	0.44	0.42	0.41	0.40	0.41
2	*4*	*0.92*	0.90	0.96	0.94	0.87	1.03	1.06	0.92	1.16

The communities are sorted by loyalty in increasing order. The first figure 6.30 and the last one 6.34 represents extreme behaviours where the loyalty similarities within the community are more marked, these effect also can be observed in the community profiles where certain products have an average spending quite differenced from the total mean. At the same time, the products more sensitive to the loyalty and total spending, are the products with central nodes that act like a hub in the graph. Intermediate communities have a mix of behaviours and the BNs (6.31, 6.32, 6.33) reflect different node structures, although the most sensitive nodes continue acting as a hubs, but with spending values closer to the total mean.

The BNs show that the class-products that act as hubs or central nodes in the graph, are the same class-products as in table 6.10, that present highest spending variations.

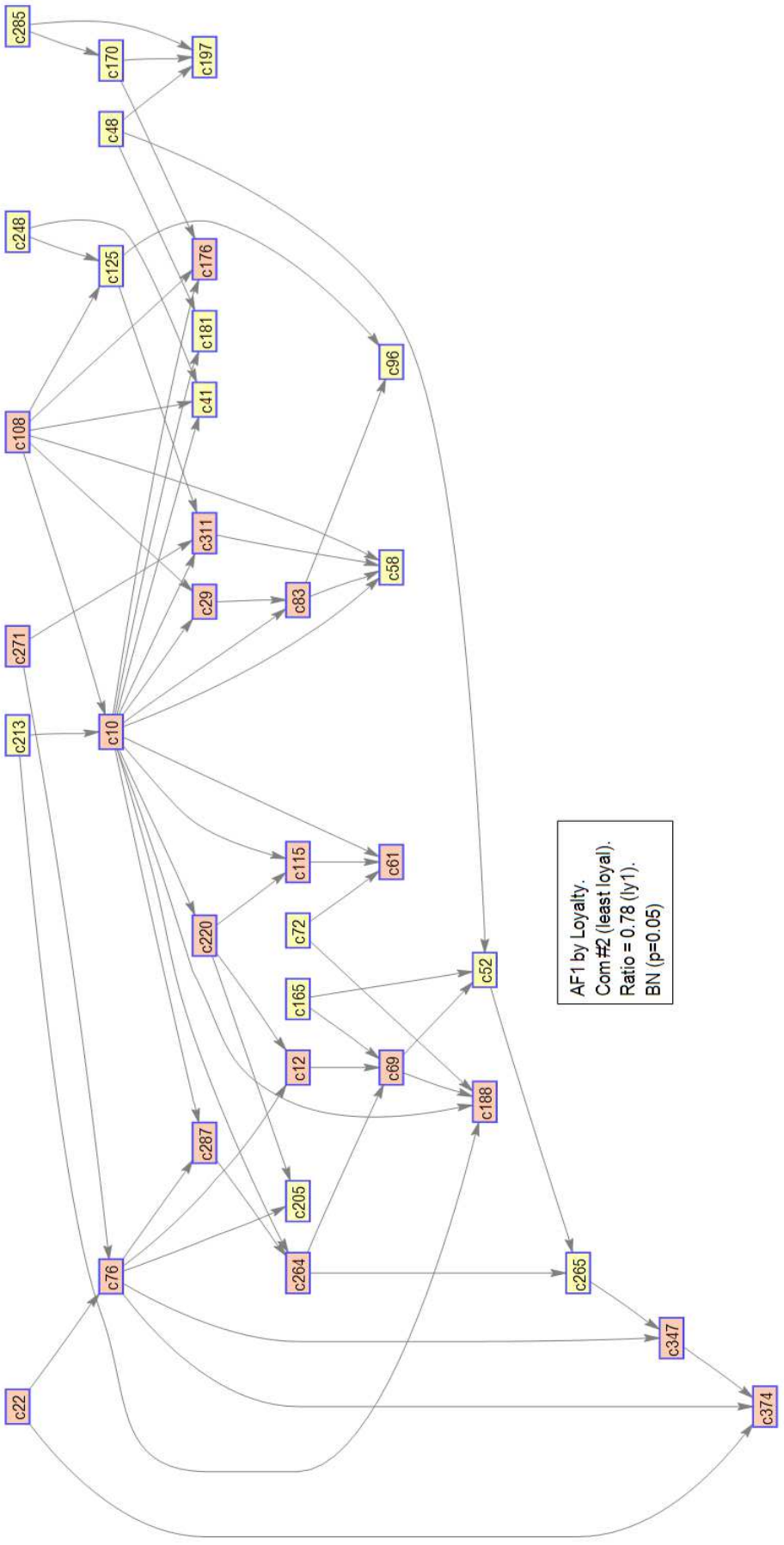


FIG. 6.30 – CI-map of least loyal community

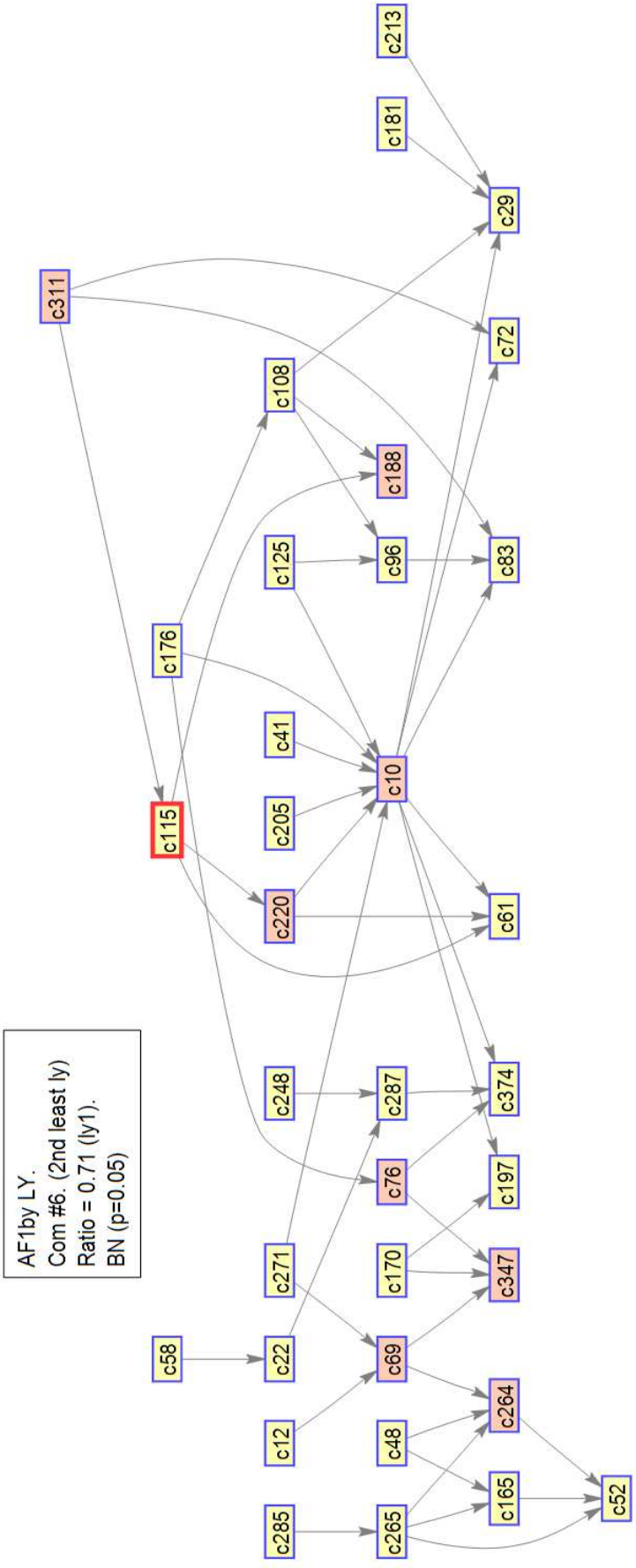


FIG. 6.31 – CI-map of 2nd least loyal community

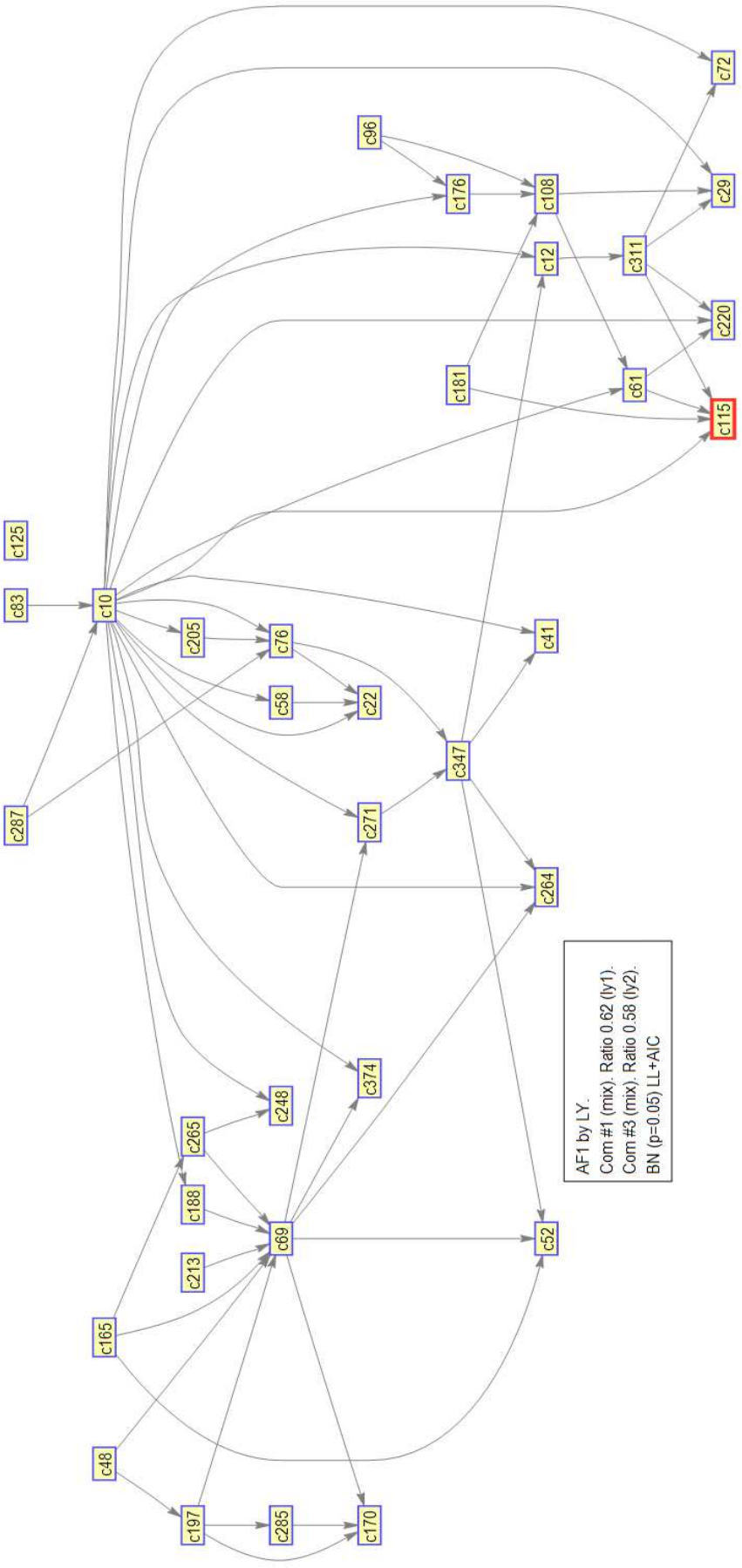


FIG. 6.32 – CI-map of middle merged loyal communities

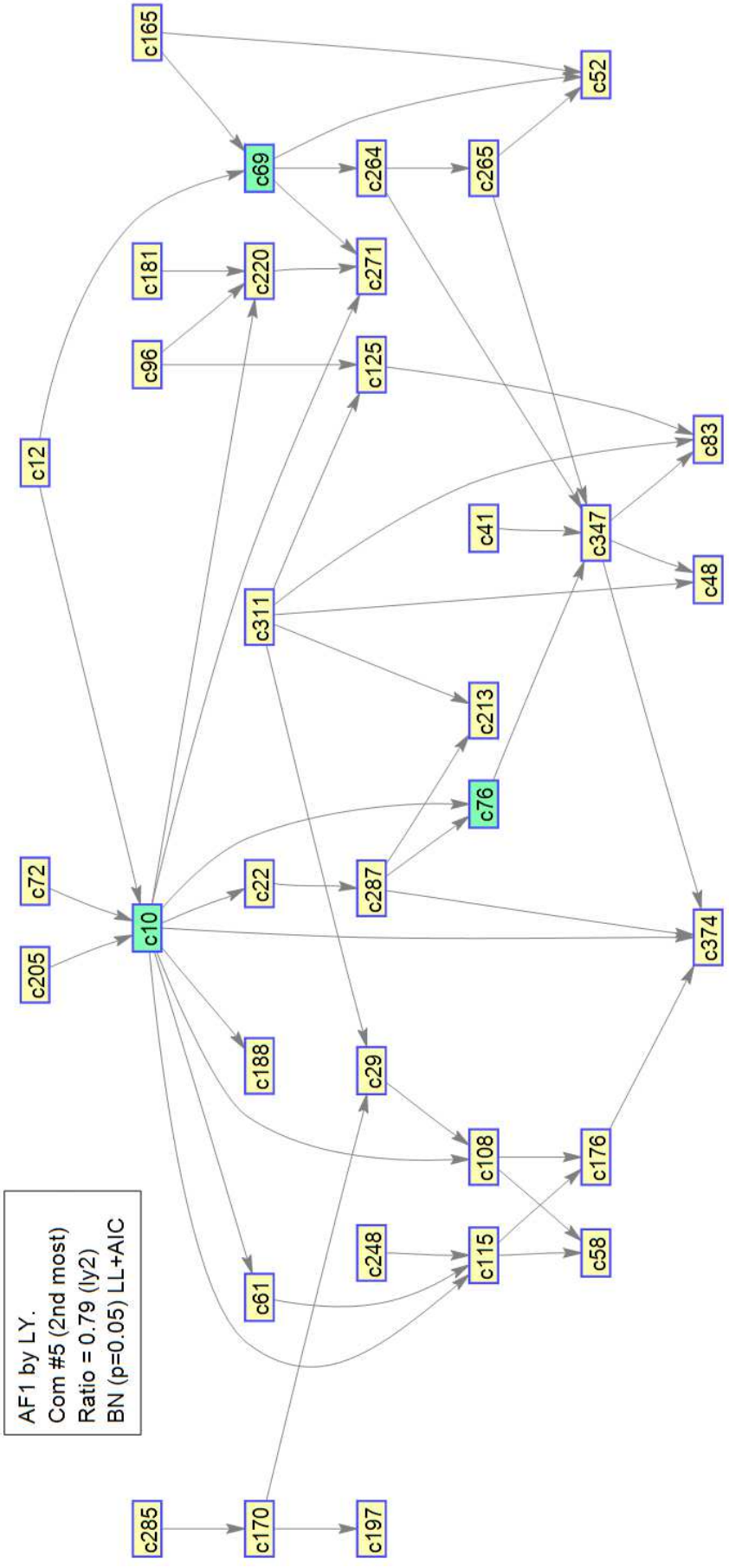


FIG. 6.33 – CI-map of 2nd most loyal community

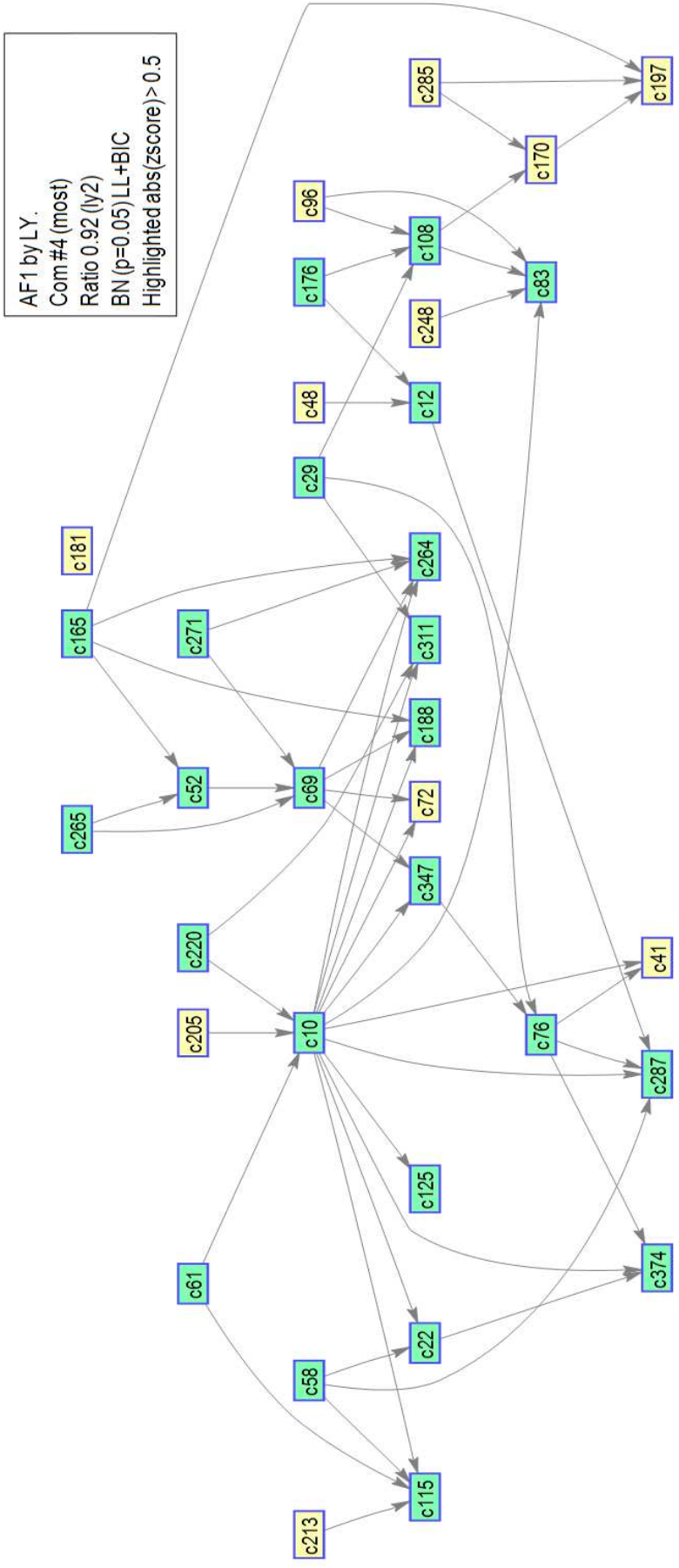


FIG. 6.34 – CI-map of most loyal community

TABLE 6.12 – External loyalty profiles with all products

Ly	c181	c197	c205	c265	c125	c10	c29	c285	c170
1	0.05	0.12	0.16	0.25	0.24	0.50	0.30	0.07	0.13
2	-0.04	-0.11	-0.15	-0.23	-0.23	-0.50	-0.30	-0.07	-0.12
3	-0.09	-0.09	-0.19	-0.23	-0.21	-0.50	-0.20	-0.11	-0.11
4	-0.13	-0.16	0.05	-0.20	-0.17	-0.40	-0.10	0.09	-0.11
Ly	**c48**	**c76**	**c165**	**c52**	**c347**	**c69**	**c248**	**c96**	**c108**
1	0.12	0.44	0.19	0.19	0.39	0.45	0.08	0.12	0.35
2	-0.10	-0.40	-0.17	-0.20	-0.36	-0.40	-0.07	-0.10	-0.32
3	-0.10	-0.40	-0.19	-0.20	-0.40	-0.40	-0.11	-0.10	-0.35
4	-0.20	-0.50	-0.23	-0.20	-0.31	-0.50	-0.04	0.00	-0.22
Ly	**c213**	**c41**	**c72**	**c58**	**c176**	**c12**	**c83**	**c22**	**c271**
1	0.13	0.19	0.17	0.24	0.29	0.30	0.33	0.29	0.33
2	-0.12	-0.17	-0.16	-0.23	-0.30	-0.30	-0.30	-0.26	-0.30
3	-0.12	-0.15	-0.15	-0.12	-0.20	-0.30	-0.28	-0.31	-0.40
4	-0.20	-0.21	-0.20	-0.12	-0.30	-0.30	-0.30	-0.25	-0.30
Ly	**c61**	**c220**	**c374**	**c188**	**c115**	**c264**	**c287**	**c311**	
1	0.33	0.37	0.36	0.34	0.39	0.38	0.39	0.42	
2	-0.30	-0.35	-0.30	-0.31	-0.40	-0.36	-0.40	-0.40	
3	-0.30	-0.28	-0.40	-0.25	-0.30	-0.32	-0.40	-0.32	
4	-0.30	-0.34	-0.30	-0.40	-0.40	-0.42	-0.40	-0.29	

Turning the attention to the class label profiles of the products more sensitive, table 6.13 shows their total spend is closer to the whole population than the spend of the community profiles. In these products, there is a discrepancy between the negative spend of the community profiles associated with *Ly1* (communities 2, 6 and 1 in table 6.10) and the positive spend of the external profile of *Ly1*. The opposite situation happens for the profiles associated to *Ly2*. For instance, community 4 has 92% of customers in the *Ly2* category, however the community average spending profile is predominantly positive from table 6.10, but the *Ly2* spending profile from table 6.13 is predominantly negative. This is a consequence of the noise in the data, the class labels are very mixed, but the manifold communities are able to extract shopping basket similarities to give more pure behaviours, mitigating the noise.

TABLE 6.13 – Class labels profiles of products most sensitive to FIN loyalty

Ly	c10	c76	c347	c69	c115
1	0.5	0.44	0.391	0.45	0.39
2	-0.5	-0.4	-0.36	-0.4	-0.4
3	-0.5	-0.4	-0.4	-0.4	-0.3
4	-0.4	-0.5	-0.31	-0.5	-0.4

For instance, looking at the spend distributions of the class-products c10 and c115 in the figures 6.35 and 6.36, one may observe how the distributions are displaced to the right (increased spend) as the community is more set to the right of the manifold. There are many observations (customers) in the shopping baskets with zero values, meaning they have not bought certain products. In these plots, the percentage of customers that do not buy a product is estimated by looking at the Y-axis values where X-axis is close to zero. For the c10 product, there is approximately 85%, 62%, 40%, 19%, 8% and 1% of customers for communities 2, 6, 1, 3, 5 and 4 respectively that do not buy product c10. A similar pattern emerges for the c115 figure but with higher percentages.

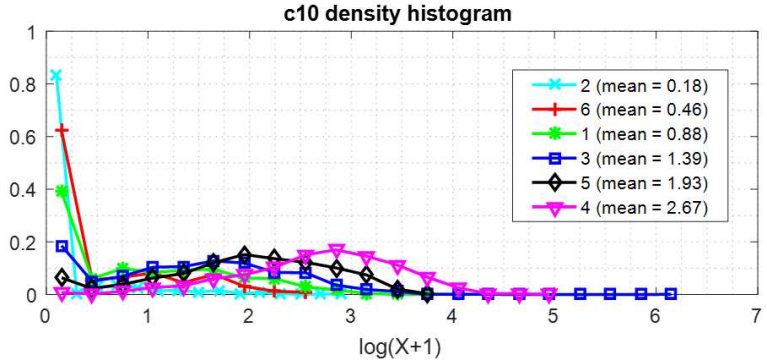


FIG. 6.35 – Community spending of product c10

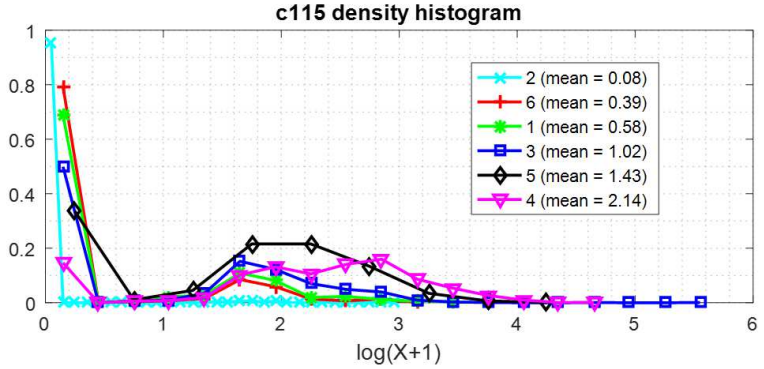


FIG. 6.36 – Community spending of product c115

6.2.11 Conclusion of retail case study

In this section, the FIN pipeline has been applied to customer shopping baskets, where the noise associated with real data has completely determined the structure of the manifold, the poor MLP performance reflecting the main constraint to the case study. The analysis of a customer data sample of aggregated shopping basket data resulted in the following insights:

1. The shopping basket data organised according to a probabilistic model of *loyalty* classes sits on an approximately linear manifold. This means that the intrinsic noise has linearised the neural network model to the point where it is closely equivalent to using logistic regression, for which the manifold generated by the Fisher Information metric is 1-dimensional and strictly equal to the risk score $\beta \times \mathbf{X}$. Therefore, loyalty segments can be inferred from the risk score using clustering methods or accepted statistical methods such as the log-rank test.

2. The segmentation obtained by the manifold clustering is more specific than the original labels, in the sense that the ratio of mean expenditure per category between segments is higher.

3. Indicator variables that are in the adjacency set of loyalty table 6.10, are also highly connected (hub nodes) in segment specific CI-maps 6.2.10.2. These variables represent categories that are proxies for shopping profiles associated with loyalty. They are likely related to underlying latent variables.

4. These indicator variables 6.10 show the strongest variation in spending between inferred loyalty segments.

As future work, a further research in the methods to reduce the impact of the noise of the real-world data in the FIN procedure is necessary.

6.3 Music data study case

This section is focused on the application of the FIN work-flow into a sample of the Million Song Dataset (MSD) [27]. Under the scope of Music Information Retrieval (MIR), the data analyses the spectral features of multiple songs which are classified (labelled) by music genre.

6.3.1 Introduction

The choice of this dataset for the Fisher manifold analysis is for several reasons, first of all it represents a good example for the kind of situations where the class labels do not correspond exactly with the features information, i.e. the songs are represented objectively by the spectral features but also the data is labelled by genre, a set of labels defined subjectively by artist style, album tendencies or other factors that mean the songs may not exactly match with the allocated musical genre. The most common genres have been selected (pop-rock, rap, and jazz), with the expectation of a smooth

transition in the song distributions across different genre regions, essentially due to the great variety of sub-genres or styles.

Another reason is that, up to this point, the Fisher manifolds of real cases presented a 1-dimensional linear behaviour (aneurysm and exotic particles datasets), due to the binary classifier and the noise. As in this chapter a dataset with three different class labels (genres) are introduced, a Fisher manifold with higher dimensionality than the 1-dimensional case is expected, although any significant noise in the data will tend to produce the opposite effect.

One interesting aspect of this dataset is that one can listen to the songs and compare how they are distributed in the Fisher manifold according the metric defined by the MLP that has only used the music spectral features as input.

In this chapter the Fisher manifold will present density variations, providing a good opportunity to use of the PQC developed in the previous Chapter 4 to find communities in the embedded Fisher manifold.

6.3.2 Methodology

The methodology of this case study differs from the retail study because in this case the feature associations are not analysed, since it has no interest for this study. The methodology can be defined in the following steps:

1. Select a random sample of the Million Song dataset, with genre labels well balanced.
2. Select the features set with the best trade-off between MLP accuracy, features number and the amount of genre labels.
3. Compute the Fisher metric with the selected MLP model and obtain the local pairwise distances with the Spark implementation.
4. Compute the global pairwise distance with Floyd-Warshall algorithm (APSP) in Matlab, the outcome is the Fisher manifold defined by the adjacency matrix of pairwise distances.
5. Apply the spectral clustering to obtain communities from the similarity matrix, the outcome is a set of community labels.
6. Apply a Euclidean embedding of the Fisher manifold and analyse the manifold density distribution.

7. Apply the PQC in the embedded Fisher manifold and obtain the cluster labels.
8. Compare labels results between both clustering methods.
9. Analyse the clusters patterns and how they are distributed in the embedded Fisher manifold.
10. Show the location of some famous songs in the manifold.

6.3.3 Data description and feature selection

The features of this dataset have been obtained from the Information Management and Preservation Lab, at the Department of Software Technology and Interactive Systems at Vienna University of Technology. This laboratory has used the MSD for benchmarking spectral features of the MIR domain, having many features sets in their MSD benchmarks website [1].

With so many features there is the question of choosing a feature set for the MLP training and how many genres to sample. To answer to these questions, the MLP performance has been used as guidance, choosing the lowest dimensionality possible with a high MLP accuracy. The results have been assessed for different pre-processing of the data: no normalization, standard normalization (z-score) and row-normalization, with the standard normalization providing the best results.

Table 6.14 shows the MLP and LR accuracy on the test set (15% sample size) for the different features sets on the MIR domain. For the MLP and the LR has been used the Matlab built-in functions from the Deep Learning toolbox [97].

Originally, the data contained 13 different genres, for this study only 3 genres have been used (keeping the same samples per genre). In addition, the table shows the accuracy of the logistic regression (LR), where linearity can be checked if MLP and LR have similar performance. The main reason LR is successful is due to the high level of noise in the dataset, where a simple linear discriminator is enough to separate most of the genres.

With regard to MLP accuracy where there are almost 5K observations with roughly 1600 songs per genre, the best trade-off between MLP accuracy and dimensionality is obtained using the 16 *Low-level features* giving 72.5% accuracy. It is important to recall that an MLP performance of at least 60% accuracy is critical for the pipeline, as the Fisher metric will rely on the information contained in the classifier model. The accuracy is the Fisher metric the accuracy is the best indicator to check if the metric associated with the MLP probabilities can map the input space with the appropriate class labels.

[1] `http://www.ifs.tuwien.ac.at/mir/msd/download.html`

TABLE 6.14 – MLP performance on different feature sets

Features set	**Dim**	**MLP acc. (%)**		**LR acc. (%)**
		13 genres	**3 genres**	**3 genres**
Rhythm histogram	60	28.4	60.0	60.8
Statistical Spectrum Descriptors	168	41.1	73.7	77.3
Area moments	20	20.5	54.4	55.0
MFCC	26	34.6	67.3	69.6
Low-level features	***16***	***31.8***	***72.5***	***70.3***
Low-level features Derivatives	96	36.7	71.3	76.5
LPC	20	29.9	66.2	64.6
Moment Methods	20	27.0	64.3	64.6

A more detailed MLP performance is depicted in figure 6.37, where the results (accuracy, sensitivity, specificity, PPV and NPV) are presented following the format of the table 3.2. In the figure, the genres correspond with the following numbers: Rap is 1, Pop-rock is 2 and Jazz is 3.

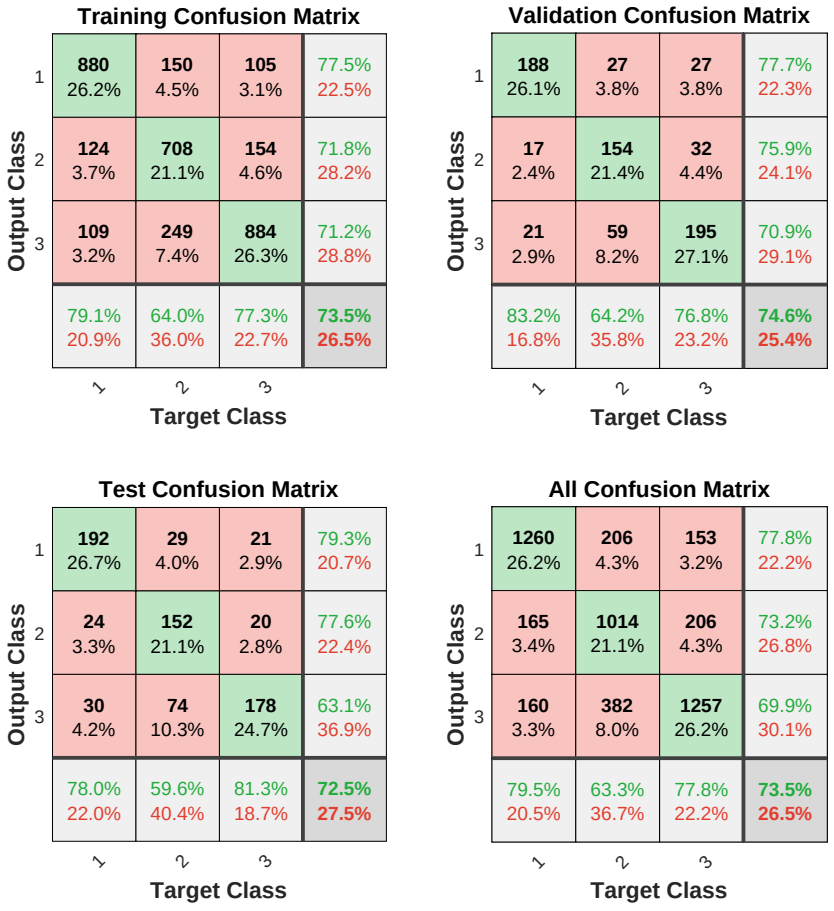


FIG. 6.37 – Music MLP performance for genre labels

The description of the spectral low-level features is shown in the table 6.15, where there are two main types: based on the standard deviation ($[X1, X8]$) or on average features ($[X9, X16]$).

TABLE 6.15 – Music spectral low-level features

Feature	**Names of Low-Level features**
X1	Spectral Centroid std
X2	Spectral Rolloff Point std
X3	Spectral Flux std
X4	Compactness std
X5	Spectral Variability std
X6	Root Mean Square std
X7	Fraction Of Low Energy Windows std
X8	Zero Crossings std
X9	Spectral Centroid mean
X10	Spectral Rolloff Point mean
X11	Spectral Flux mean
X12	Compactness mean
X13	Spectral Variability mean
X14	Root Mean Square mean
X15	Fraction Of Low Energy Windows mean
X16	Zero Crossings mean

6.3.4 Pipeline results

Once the MLP is trained and the Fisher pairwise distances computed, the Fisher manifold structure under the Euclidean embedding can be analysed. The next subsection starts with the Sammon mapping and the cMDS embedding.

6.3.4.1 Manifold structure with cMDS

Firstly the embedding with the Sammon mapping is discussed, figure 6.38 where the colours represent the genres with which the songs were originally labelled. The Sammon mapping tends to not preserve global distances and in this case one may observe there are some outliers with distorted distances, reflecting the fact that Sammon mapping is less reliable to representing density distributions than the cMDS embedding. And for this reason this chapter will focus on the cMDS results.

Figure 6.39 shows the accumulated eigenvalues of the Fisher manifold embedded with cMDS. The eigenvalues indicate that the Fisher manifold is bi-dimensional, that means the MLP needs 2 dimensions to discriminate the genres. This low-dimensionality is

partly due to the inherent noise in the data, that makes linear boundaries equally efficient than a non-linear MLP, as evidenced in the LR performance in table 6.14.

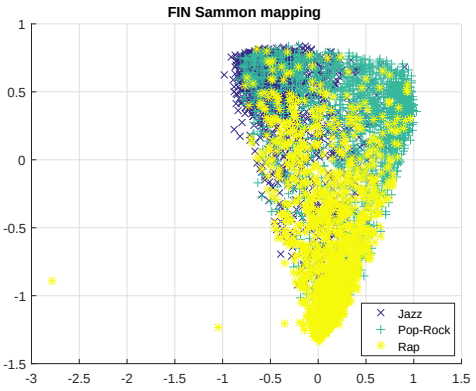


FIG. 6.38 – Music FIN Sammon mapping

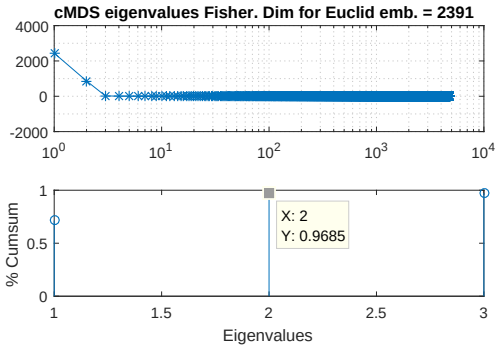


FIG. 6.39 – Music FIN cMDS eigenvalues

Figure 6.40 presents the two main eigenvectors of the FIN cMDS embedding, where the significant mixing of genre is evidence of the noise in the data. However, observing the MLP predictions in figure 6.41, they depict simple boundaries separating the genres in approximately equal areas of the Fisher manifold, and is why a LR has similar results.

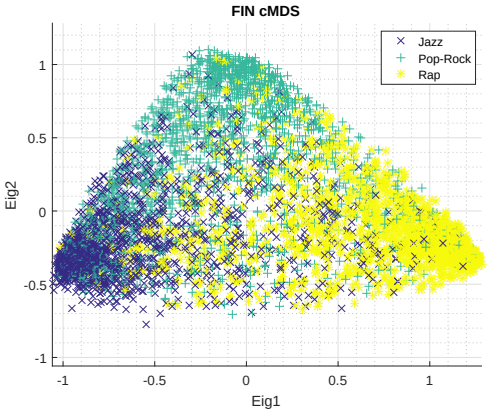


FIG. 6.40 – Music FIN cMDS by genres

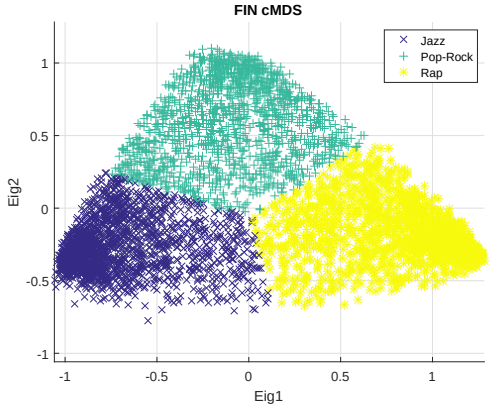


FIG. 6.41 – Music FIN cMDS by MLP predictions

By applying the spectral clustering with Newman's algorithm and using the heuristic method for the length scale, five communities were recovered shown in figure 6.42. They segment the manifold into five regions that do not appear to correspond with the peak density distributions. If one observes figure 6.43 with the bi-dimensional histogram, the density peaks are centred near the extremes of the Fisher manifold where the genre concentration is most pure. This effect can be evidenced in the legend of the communities' plot, that shows the ratio of maximum genre membership per community, a kind of genre prevalence per community. Communities 1, 2 and 3 have ratios of 0.80, 0.79,

0.84 respectively, compared with communities 4 and 5 which are in the middle of the manifold that have ratios of 0.47 and 0.42.

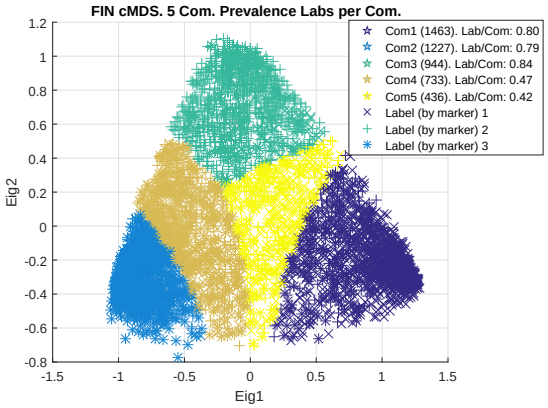


FIG. 6.42 – Music FIN cMDS by Newman's communities

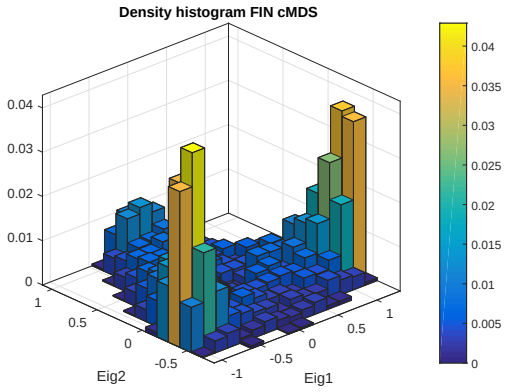


FIG. 6.43 – Music FIN cMDS 2D histogram

6.3.4.2 cMDS with Euclidean distances

Just for completeness, figures 6.44 and 6.45 depict the cMDS embedding of the manifold based on Euclidean pairwise distances. The eigenvalues cumulative sum indicates that at least four or five principal components are needed to retain more than 80% of the variance. The Euclidean cMDS representation, which is equivalent to PCA, shows that two principal components are not enough to separate the genres, as there is significant mixing of genre types.

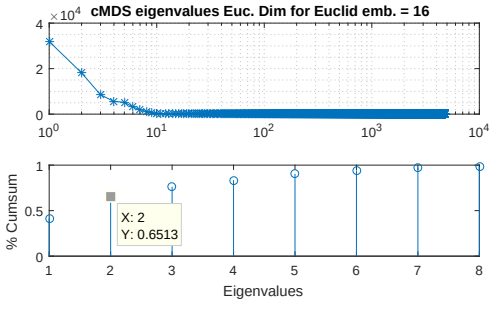


FIG. 6.44 – Music Euclidean cMDS eigenvalues

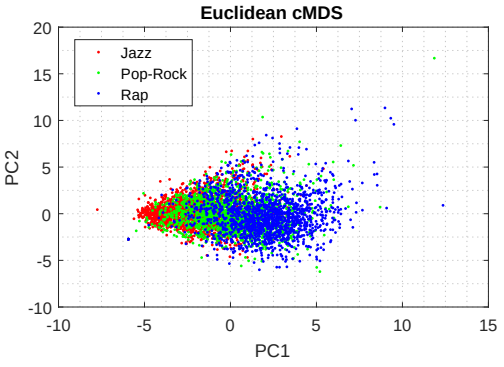


FIG. 6.45 – Music Euclidean cMDS

6.3.4.3 Community finding with PQC on cMDS embedding

If, instead of the spectral clustering, PQC was used, then the density-based method would provide us with tools for selecting the appropriate PQC hyper-parameters, minimizing the Average-Negative-Log-Likelihood of cluster membership. Figure 6.46 shows

the ANLL plot which indicates that $\sigma = 15\%knn$ and $log(E_{th}) \in [-3, -1.5]$ are good hyper-parameters for this data. Selecting these parameters a solution is obtained and presented in figure 6.48, where the $\max_K P(X|K)$ is depicted in figure 6.47, representing the maximum probability of belonging to any cluster. In the manifold, there are regions where the cluster membership is certain, but in the middle regions is not so clear anymore, showing regions with a significant mixture of genre.

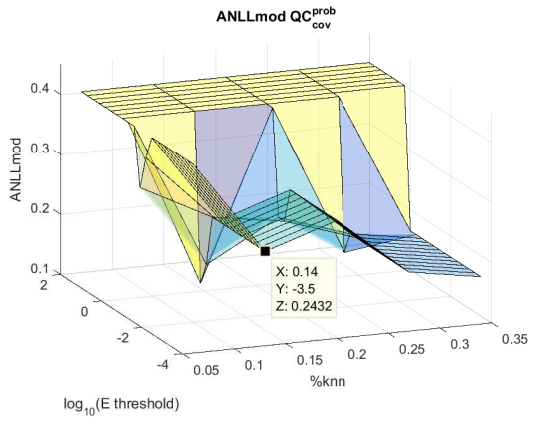


FIG. 6.46 – Music FIN cMDS PQC ANLL

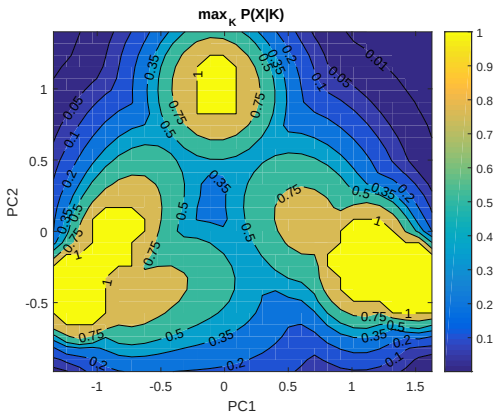


FIG. 6.47 – Music FIN cMDS PQC $\max_K P(X|K)$

The genre prevalence per community in the case of PQC clusters is depicted in figure 6.47. There are six communities, where three belong to the genre regions of higher density with a genre prevalence of 0.82, 0.85 and 0.91 for Jazz, Pop-Rock and Rap, respectively. The other three communities lie in the intermediate regions, with lower prevalences $\in [0.47, 0.47, 0.60]$.

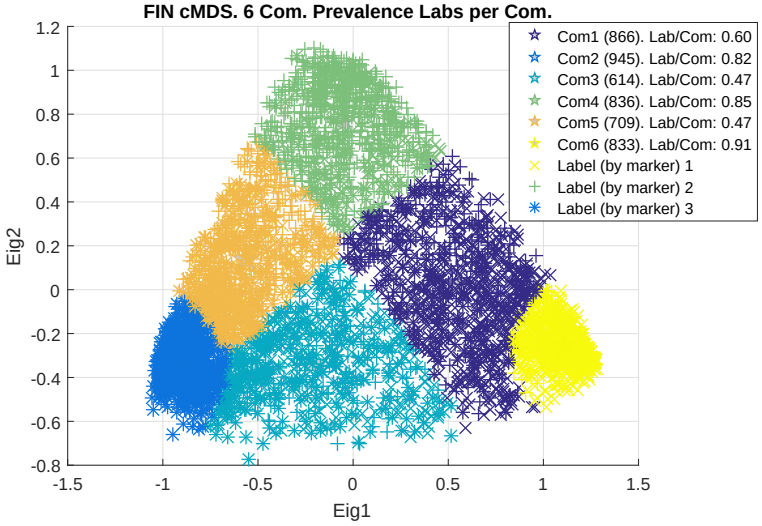


FIG. 6.48 – Music FIN cMDS by PQC clusters

6.3.4.4 Communities profiles

One interesting option for these groups is to build the communities profiles, i.e. the main feature characteristics of each community standardized with reference to the whole data, see eq. 3.26. In this case, one can highlight the most important attributes of each genre (communities of higher density) and compare them with the other communities. In addition, the main attributes of the class labels without taking into account their distribution in the Fisher manifold can be examined, too. The results are presented in tables 6.16 and 6.17, which display the features based on standard deviation and mean values respectively. Given the profiles are standardized, only those absolute values greater than 1 can be considered different enough from the overall mean of the data (significance in terms of one standard deviation).

The notation of the table for the column position is PR as Pop-Rock, J as Jazz and R as Rap. The positions refer to the cMDS embedding where Jazz is bottom left, Rap bottom right and Pop-Rock from the middle and up. Inspecting the features, Jazz and Rap are characterized by greater absolute values. Pop-Rock features being closer to zero intuitively makes sense, given that Pop-Rock includes many music styles ranging from heavy-metal to commercial-pop, and therefore producing an average value closer to zero. The intermediate communities also have lower absolute values than the extremes communities. In terms of the features, those based on standard deviations have greater prominence than the means values.

Finally, it is important to remark that, not only in this case but also in general, the profiles based on class labels tend to be close to zero, given the class labels are not clustered by similarities or feature distances, producing averaged profiles close to the whole data average.

TABLE 6.16 – Community profiles with std features

Com.	**Position**	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
1	PR & R	0.40	0.45	0.42	-0.02	0.46	0.44	-0.15	0.45
2	J	-0.89	-1.02	-1.00	0.21	-1.00	-1.00	0.32	-0.98
3	J & R	-0.06	-0.07	-0.40	0.37	-0.25	-0.23	0.59	0.00
4	PR	-0.22	-0.09	0.22	-0.46	-0.05	-0.08	-0.50	-0.25
5	J & PR	-0.45	-0.41	-0.62	-0.22	-0.63	-0.61	-0.13	-0.44
6	R	1.26	1.19	1.29	0.15	1.44	1.44	-0.03	1.26
Class labels		**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
Rap		0.73	0.69	0.71	0.13	0.83	0.81	-0.03	0.74
Pop-Rock		-0.20	-0.11	-0.06	-0.24	-0.18	-0.18	-0.20	-0.20
Jazz		-0.52	-0.56	-0.64	0.11	-0.64	-0.61	0.23	-0.53

TABLE 6.17 – Community profiles with mean features

Com	**Position**	**X9**	**X10**	**X11**	**X12**	**X13**	**X14**	**X15**	**X16**
1	PR & Rap	0.11	0.12	0.40	-0.35	0.39	0.37	-0.05	0.19
2	J	-0.74	-0.81	-1.00	0.68	-1.04	-1.05	-0.22	-0.95
3	J & R	-0.51	-0.54	-0.57	0.11	-0.53	-0.57	0.18	-0.55
4	PR	0.62	0.79	0.75	-0.34	0.80	0.90	-0.10	0.81
5	J & PR	0.01	-0.02	-0.56	0.22	-0.36	-0.33	-0.16	0.02
6	R	0.47	0.41	0.86	-0.34	0.67	0.61	0.42	0.45
Class labels		**X9**	**X10**	**X11**	**X12**	**X13**	**X14**	**X15**	**X16**
Rap		0.23	0.19	0.50	-0.30	0.42	0.37	0.21	0.22
Pop-Rock		0.21	0.31	0.20	-0.13	0.28	0.34	-0.11	0.31
Jazz		-0.43	-0.49	-0.69	0.43	-0.69	-0.69	-0.10	-0.53

6.3.4.5 Analysis of famous artists songs

In this section, songs of famous artists are mapped onto the embedded Fisher manifold to identify where they are located, the advantage of analysing songs is that they can be listened to and their label/community assessed for appropriateness of the attributions, albeit subjectively.

The following table 6.18 enumerates the songs by Id for identifying them in the next figure 6.49. The table contains the external genre and the predicted genre by the MLP.

TABLE 6.18 – List of famous songs mapped into the Fisher manifold

Id	**Lab/MLP**	**Song Artist**	**Song title**
1	J - J	Count Basie	Segue In C
2	J - J	Duke Ellington	Black And Tan Fantasy
3	R - R	Eminem	We Made You
4	J - J	Frank Sinatra	I Should Care
5	J -R	George Benson	Stairway To Love
6	J - J	Glenn Miller	Happy In Love
7	R - R	Ice Cube	A Bird In The Hand
8	J - R	Jamie Cullum	Love Aint Gonna Let You Down
9	R - R	Jay-Z	Threat
10	J - J	Juliet Roberts	Carriacou Sunrise
11	R - R	Kanye West	Flashing Lights
12	PR - PR	Korn	Politics (Claude Le Gache Edit)
13	PR - J	Led Zeppelin	Since Ive Been Loving You
14	PR - J	Little Richard	Long Tall Sally (Take 1)
15	J - J	Louis Armstrong	I Cant Give You Anything But Love
16	J - J	Louis Armstrong	Alexanders Rag Time Band
17	PR - J	Martha Wainwright	These Flowers
18	J - J	Miles Davis	Dear Old Stockholm
19	PR - PR	Naer Mataron	The Life And Death Of Europa
20	J - J	Nat King Cole	I Get A Kick Out Of You
21	PR - J	Neil Diamond	Girl Youll Be A Woman Soon
22	PR - PR	Neil Young	Revolution Blues
23	PR - PR	Pet Shop Boys	Rent (2001 Digital Remaster)
24	PR - J	Robbie Williams	Morning Sun Reprise
25	J - J	Slavic Soul Party!	Juan Colorado
26	R - PR	Snoop Dogg	Gangsta Luv
27	J - PR	The Flying Luttenbachers	Clank
28	PR - PR	The Jimi Hendrix Exp.	Hey Joe
29	PR - R	The Rolling Stones	Cherry Oh Baby
30	PR - PR	The Velvet Underground	Rock And Roll (LP Version)
31	R - PR	Vanilla Ice	Its A Party
32	R - R	Will.I.Am	Tai Arrive

Apart from the famous songs, two more songs have been included due to their closeness and rare positioning in the middle-top manifold. Their ids are 27 (Jazz) and 19 (Pop-Rock), and examined to explain why a Jazz song may be so far away from the main Jazz region (bottom-left). If the reader can listen to both songs they will discover that the Jazz song (id 27) is quite an experimental style with a sound that could be identified as industrial noise. On the other hand, the Pop-Rock song (id 19) is extreme heavy metal, surprisingly labelled as Pop-Rock.

Additional information about these songs features can be found in next Subsection 6.3.4.6.

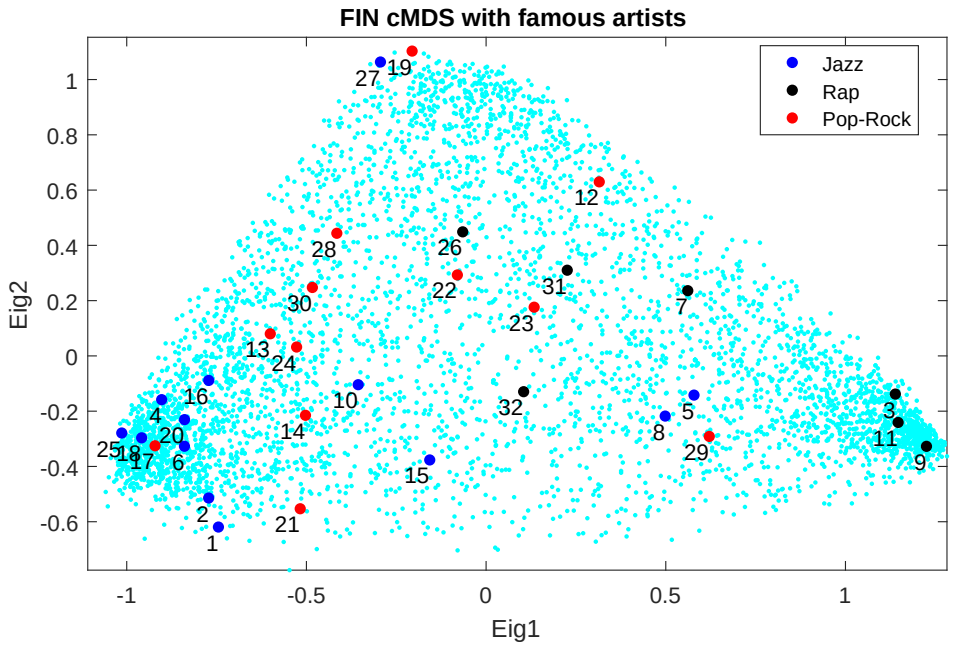


FIG. 6.49 – Particles FIN Sammon mapping by mass label

6.3.4.6 Spectral low-level features

This subsection shows additional information about the famous artist songs listed in table 6.18, it contains the features of the selected songs in the tables 6.19 and 6.20. Bear in mind, sometimes the spectral features characterize the main attributes of the wave sound but it is possible that these features are not closely linked to the layman perception of song style. On the other hand, it is remarkable the discrepancies between the genre labelling and the spectral features similarities if one observes the high noise and genre mixing in figures 6.40 and 6.49.

6.3.5 Conclusion of music case study

Probabilistic classifiers induce a local similarity metric at each location in the space of input data. This is measured by the Fisher Information Matrix. Pairwise distances in this Riemannian space are measured along geodesics. These distances are useful for generating a similarity map of the data. This work describes an application to Music Information Retrieval. A Euclidean embedding is produced for three genres in a cohort of the Million Song Dataset, which is then segmented to find songs that clearly belong to a given music genre whilst others reflect fusion styles. This validates this generic method in a particular application domain and illustrates its value for intelligent data retrieval.

TABLE 6.19 – Features std-based of famous artists

Id	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
1	-0.6	-0.6	-1.0	1.2	-0.9	-0.9	-0.5	-0.4
2	-0.2	-0.5	-1.2	0.8	-1.3	-1.2	0.7	-0.5
3	1.6	1.5	2.8	-0.3	1.6	2.1	-0.8	1.5
4	-0.4	-0.6	-1.2	0.6	-1.5	-1.6	1.4	-0.5
5	0.5	0.4	0.5	0.5	0.3	0.6	0.9	0.4
6	-0.9	-1.0	-1.0	-0.1	-0.9	-0.8	-0.1	-0.9
7	-0.5	-0.4	1.0	0.0	1.1	1.0	-1.0	-0.4
8	-0.5	-0.6	0.9	-0.4	1.1	1.1	1.5	-0.4
9	0.4	0.7	2.6	0.1	2.2	2.3	1.0	0.6
10	-0.6	-0.6	0.1	1.3	0.0	0.0	-1.1	-0.7
11	0.6	0.7	1.2	-0.5	1.5	1.6	0.3	0.7
12	-0.5	-0.2	1.4	-0.4	0.9	0.7	0.3	-0.5
13	-0.6	-0.5	-0.7	-0.3	-0.5	-0.4	-0.4	-0.4
14	0.0	-0.2	-0.9	3.0	-0.4	-0.2	1.0	-0.3
15	1.3	0.4	-1.1	-0.4	-0.9	-0.7	1.6	0.8
16	-0.6	-0.8	-1.0	-0.3	-0.7	-0.5	0.3	-0.8
17	-1.2	-1.2	-0.9	0.8	-0.7	-0.7	0.0	-1.2
18	-0.8	-0.9	-1.0	0.2	-1.0	-0.9	1.4	-0.8
19	0.0	-0.5	-0.3	-0.7	-0.6	-0.7	-1.3	-0.5
20	-0.5	-0.6	-1.0	-0.1	-0.9	-0.8	0.8	-0.6
21	0.2	-0.1	-1.2	0.3	-1.5	-1.5	0.9	0.2
22	-0.5	-0.4	0.4	0.1	0.3	0.5	-0.6	-0.5
23	-0.4	0.0	0.1	0.9	0.0	0.1	-0.7	-0.2
24	-0.7	-0.8	-0.2	-0.2	-0.4	-0.3	-0.1	-0.7
25	-1.0	-1.2	-1.1	1.1	-1.3	-1.4	0.3	-1.3
26	-0.9	-0.9	0.2	-0.3	0.1	0.1	-0.9	-0.7
27	-0.4	-1.3	-0.2	-0.8	-0.8	-0.7	-0.4	-1.1
28	-0.6	-0.3	-0.5	-1.1	-0.5	-0.5	-0.6	-0.6
29	1.8	1.9	-0.8	-0.8	-0.5	-0.3	1.3	1.9
30	0.3	0.4	-1.0	-0.1	-1.1	-1.0	-0.6	0.2
31	1.4	1.5	-0.8	0.0	-0.7	-0.7	-1.0	1.4
32	0.2	0.7	0.6	-0.1	0.5	0.7	2.4	0.3

This work has presented a MIR application of Fisher manifolds. The goal is to arrange and cluster the songs with human-labelled genre information contained in the MLP. The similarities between songs have been based on spectral wave sound features, purely objective measures without any subjective human-perception whatsoever. The result is a Fisher manifold where the genre labels are somewhat mixed but there are some high-density regions in the cMDS embedding where there is a predominant genre. PQC has been used for community finding applied to the embedded Fisher manifold. The obtained communities (clusters) classify the songs into two main kinds of clusters, namely, those belonging to a pure genre and others that belong to an intermediate region, characteristic of a fusion of styles. Finally, the work analyses a list of famous songs mapped

Table 6.20 – Features mean-based of famous artists

Id	X9	X10	X11	X12	X13	X14	X15	X16
1	-1.1	-1.0	-1.1	0.9	-1.2	-1.3	1.0	-1.1
2	-0.5	-0.6	-1.2	0.5	-1.6	-1.6	-0.4	-0.9
3	1.9	1.8	2.3	-0.5	1.2	1.3	0.2	1.9
4	0.5	0.4	-1.2	0.4	-1.7	-1.7	-0.8	0.1
5	0.0	-0.1	0.1	0.5	0.4	0.4	0.2	0.1
6	-1.0	-1.0	-1.0	0.2	-0.7	-0.7	0.1	-1.3
7	-0.7	-0.6	1.4	-0.2	1.6	1.5	-0.6	-0.5
8	-0.9	-0.9	0.5	-0.6	0.8	0.7	-0.1	-1.0
9	-0.6	-0.5	1.8	-1.0	1.7	1.5	1.3	-0.4
10	-0.3	-0.2	-0.1	1.1	0.2	0.2	-0.1	-0.5
11	-0.3	-0.3	1.1	-0.8	1.3	1.2	0.6	-0.1
12	0.1	0.3	1.2	-0.7	1.3	1.4	0.8	0.5
13	-0.3	-0.2	-0.5	0.1	-0.2	-0.2	0.0	-0.3
14	0.3	0.2	-0.7	1.4	0.0	0.1	-1.0	-0.1
15	0.4	0.1	-1.1	0.0	-1.2	-1.2	0.6	0.2
16	0.2	0.0	-0.9	-0.1	-0.8	-0.7	-0.3	-0.3
17	-1.3	-1.2	-0.7	1.4	-0.3	-0.3	-0.2	-1.3
18	-0.3	-0.4	-1.1	1.3	-0.9	-0.9	-1.0	-0.4
19	2.7	3.5	1.1	-0.8	1.0	1.2	-0.4	2.6
20	-0.1	-0.2	-1.0	0.3	-1.1	-1.1	0.3	-0.2
21	-0.8	-0.9	-1.2	-0.2	-1.7	-1.7	-0.2	-1.1
22	0.0	0.0	0.3	0.3	0.3	0.4	0.3	0.1
23	-0.2	0.0	0.2	0.2	0.5	0.6	-0.1	-0.2
24	-0.7	-0.8	-0.3	-0.2	-0.1	0.0	0.7	-0.5
25	0.1	-0.1	-1.1	2.4	-0.9	-0.9	-0.5	-0.4
26	-0.9	-1.0	0.9	-0.5	1.1	1.1	-1.1	-0.9
27	3.8	2.6	0.5	-0.5	0.6	0.9	-0.5	2.7
28	-0.4	-0.3	-0.3	-0.3	0.2	0.3	-0.1	-0.2
29	0.9	0.7	-1.0	-0.2	-1.1	-1.1	0.5	0.8
30	0.6	0.8	-0.9	0.4	-0.9	-0.8	-0.6	0.7
31	1.7	1.9	-0.7	0.0	-0.6	-0.6	-0.5	1.8
32	0.4	0.3	-0.3	0.6	-0.4	-0.4	1.5	0.6

into the Fisher manifold in order to illustrate the achieved results and facilitate their interpretation.

6.4 Conclusion of complete framework

This chapter has applied the complete framework to two completely different case studies. The first part has focused on describing the procedure to obtain the Fisher manifold and on comparing the two methods of clustering suggested in this thesis. The pipeline describes two possible clustering paths, but in the comparison of this chapter, it has

been seen that both paths do not have to be exclusive. The path of spectral clustering based on similarity networks is good for segmentation tasks and the path of PQC is good for density discrimination, but in general there is not a huge difference in the labels of both methods, being quite consistent between them. The level of matching has been measured with the Cramer's V statistic and for the both case studies the score has been similar: $C_V\left(communities, clusters_{PQC}\right) \approx 0.77$. In any case, the label differences in both methods will be accentuated when the Fisher manifold distribution presents high density variations.

The second part of the chapter has focused on analysing in detail the two case studies, where the presence of noise in the data has been critical to determine the shape of the manifold, making the MLP probabilistic model have a linear behaviour, with results similar to a logistic regression. For the first case, where the manifold was based on customer shopping baskets, the noise and the fact that the MLP model only detected two labels produced that the shape of the manifold was one-dimensional. The music study case would have had a similar effect, but the fact that the MLP model detected the three genre labels has made the Fisher manifold two-dimensional. For the music case, the differences in density have made the PQC more appropriate to detect the clusters, besides mapping the manifold with the probability of belonging to any cluster, useful for identifying regions with fusion of musical genres.

Finally, the CI-maps analysis only has been applied in the retail data, where the most important products of the shopping baskets in relation to the loyalty labels have been identified. These products play the role of central nodes in the Bayesian networks, and many secondary products are connected to them. The communities at both ends of the one-dimensional manifold have more differentiated CI-maps, and the communities of intermediate regions have a transitional behaviour between both extremes.

Chapter 7

Final conclusions

This thesis has covered algorithms that belong to completely different areas with an overarching theme of developing a process to restructure data, based on the question (label) being asked. The first part of metric learning could be catalogued as semi-supervised algorithms, and the other two of clustering and structure learning as unsupervised algorithms. The common bond is the analysis of the data structure through three different perspectives, summarized as follows:

1. The structure of a manifold, which contains the relevant information and the complexity of a classifier model. The core of the algorithm is the Fisher metric applied to the input space instead of the parameter space. This local metric implicitly measures the amount of the model information in that region, where the Euclidean pairwise distances are transformed into a Riemannian manifold.

2. The structure of data density, through the bounding of nearest neighbours until they conform to a cluster. The density estimator is based on a convex potential derived from quantum mechanics. Essentially, the potential transforms the problem of density hill climbing into a gradient descent problem looking for local minima.

3. The structure of feature associations, where these associations are based on conditional independence tests. In the graph notation, the features are represented by nodes and the associations by edges. The CI-map represents an undirected graph, where the edges can be oriented to perform a direct acyclic graph as a Bayesian network. Learning the structure from the data will not provide a unique DAG solution. From the PDAG obtained through the PC-algorithm, a systematic node-ordering methodology has been developed to build a DAG, which is robust, reliable and reproducible for creating a good candidate as a Bayesian network.

These structure analyses are linked together through the pipeline described in figure 1.1, where the Fisher manifold changes the metric and redefines the neighbourhood of observations, forming a new density distribution. This new structure can be analysed with the PQC, that detects clusters by density variations. Until this point, the structure analysis is focused on similarities and relationships between observations, however the structure analysis can also focus on the relationships between features, which is the purpose of the CI-maps methodology, but taking the advantage of the segmentation provided by the clustering analysis.

After all the experiments carried on each chapter, the following conclusions can be made:

Fisher manifold

By constructing a structure of the Fisher manifold, it will contain the complexity of the MLP classifier, in such a way that the manifold dimensionality mainly depends on 1) the structure of the class labels distribution and 2) the classifier ability to discriminate them according to the available input data, i.e. the model performance.

Regarding the label distributions, different examples have been covered. For instance the case of exotic particles, where the input space had high dimensionality and the binary classifier only required a 1-dimensional Fisher manifold to discriminate the labels. As a counterexample, analysed the concentric spirals dataset, a 2-dimensional input space with a very structured label distribution. In this case the Fisher manifold needed a 3-dimensional space to discriminate the labels, with a higher dimensionality than in the input space. In general, the Fisher manifold tends to create a Riemannian space with as low dimensionality as possible.

The other factor is the model performance, which directly depends on the information contained in the input space for classifying the labels. A noisy dataset, which is the case in most of real-world data, tends to linearise and smooth the Fisher manifold, but if the noise is too high or the MLP accuracy is too low, the manifold loses structure and tends to a homogeneous distribution of the labels. As examples, there were three cases of increasingly noisy data, the music data and the shopping baskets with loyalty labels, which displayed considerable overlapping of labels in the intermediate regions of the manifold. The extreme case of poor MLP performance with the shopping baskets applied to affluence, life-style or life-stage labels. Future work in this direction could look at noise reduction in real-world scenarios such as the customer shopping baskets to mitigate the impact in the Fisher manifold.

In terms of the Fisher pipeline implementation, Spark big-data tools have been developed to reduce the runtime of the pairwise distance computations. However, the main limitation is that the algorithm does not scale enough for fairly large sample sizes, for

instance 15K observations for a medium cluster took approximately five hours runtime. As future work, it would be worth implementing the pipeline in GPGPU, where there is also a great level of parallelization.

Probabilistic Quantum Clustering

The main achievement of the PQC has been the development of an unsupervised score to assess the model hyper-parameters, where the cluster solutions are highly correlated with the supervised Jaccard score. The unsupervised score measures the average negative log-likelihood of cluster membership under a probabilistic framework, where the framework also allows for outlier detection and cluster discrimination by local density variations.

The algorithm has been developed in relatively low dimensional datasets (less than 10 dimensions), highly structured with non-spherical cluster distributions, with heteroscedasticity and local density variations. In all cases the results been better or comparable with the original quantum clustering performance.

The main limitation of any version of quantum clustering is in relation to the measurement of relative Euclidean distances in high dimensional spaces, this problem is inherent in the Gaussian kernels which have been used as density estimators. Research into different types of kernels applied to quantum clustering could be an interesting future direction for the work, taking into account that QC needs a kernel easily differentiable up to at least second order. An alternative future work could look into reducing the PQC runtime using GPGPU processing.

Conditional Independence maps

In terms of the structure learning field, for constraint-based algorithms a stable methodology has been improved for building CI-maps and their derived BNs. The main developments have been 1) the tuning of the PC-algorithm policies to reduce the structure errors with unknown data and the overall false positive error proliferation; and 2) the impact of the node ordering in the DAG construction. The process have been validated on several benchmark data, and the brain tumour application reveals congruent associations of metabolites and the tumour types. Another relevant application has been for customer profiles based on shopping baskets, where the product associations differ depending on the loyalty community.

The main limitation of the structure learning algorithms is that it is very hard to know if the structure recovered from the data is really the true structure, this limitation is aggravated in cases of data with considerable noise.

One possible line of future work would be the application of the probabilistic models generated by the BNs with the parameters learnt from the shopping baskets data. In

such a way that, for instance, the probability that a customer will buy a certain product given that other products have already been purchased can be estimated.

As final comments, I would like to highlight that, in the scope of real applications, sometimes the use of complex non-linear algorithms may be unnecessary, where the noise of real data or the scalability issues in big data environments tend to favour simpler, more interpretable and faster linear models. However, from the theoretical point of view, complex algorithms like the Fisher manifold or the PQC are still very interesting, because maybe, one day the computing power of new technologies allows a more extensive use of these complex algorithms in real scenarios. Analogous to the case of the neural networks resurgence with the deep learning paradigm being witnessed at the moment.

Bibliography

[1] J. B. Tenenbaum, V. De Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, science 290 (5500) (2000) 2319–2323.

[2] H. Ruiz, Fisher networks: A principled approach to retrieval-based classification, Ph.D. thesis, Liverpool John Moores University (2013).

[3] H. Ruiz, S. Ortega-Martorell, I. H. Jarman, J. D. Martín-Guerrero, P. J. Lisboa, Constructing similarity networks using the Fisher information metric., in: ESANN, Citeseer, 2012, pp. 191–196.

[4] H. Ruiz, I. H. Jarman, J. D. Martín, P. J. Lisboa, The role of Fisher information in primary data space for neighbourhood mapping., in: ESANN, 2011, pp. 381–386.

[5] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on, Ieee, 2010, pp. 1–10.

[6] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, The Journal of Machine Learning Research 17 (1) (2016) 1235–1241.

[7] D. Bacciu, P. J. Lisboa, A. Sperduti, T. Villmann, Probabilistic Modeling in Machine Learning, in: Springer Handbook of Computational Intelligence, Springer, 2015, pp. 545–575.

[8] P. Spirtes, C. N. Glymour, R. Scheines, Causation, prediction, and search, MIT press, 2000.

[9] S. Kullback, Information theory and statistics, Courier Corporation, 1997.

[10] H. Ruiz, T. A. Etchells, I. H. Jarman, J. D. Martín, P. J. Lisboa, A principled approach to network-based classification and data representation, Neurocomputing 112 (2013) 79–91.

[11] MATLAB, version 8.5.0.197613 (R2015a), The Mathworks, Inc., Natick, Massachusetts (2015).

[12] M. E. Newman, Modularity and community structure in networks, Proceedings of the national academy of sciences 103 (23) (2006) 8577–8582.

[13] J. W. Sammon, A nonlinear mapping for data structure analysis, IEEE Transactions on computers 18 (5) (1969) 401–409.

[14] Y. Zhao, S. Tasoulis, T. Roos, Manifold visualization via short walks, in: Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers, Eurographics Association, 2016, pp. 85–89.

[15] S. Roweis, L. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 290 (5500) (2000) 2323–2326.

[16] T. F. Cox, M. A. Cox, Multidimensional scaling, CRC press, 2000.

[17] L. v. d. Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (Nov) (2008) 2579–2605.

[18] S. J. Chambers, I. H. Jarman, T. A. Etchells, P. J. G. Lisboa, Inference of number of prototypes with a framework approach to K-means clustering, International Journal of Biomedical Engineering and Technology 13 (4) (2013) 323–340.

[19] D. Horn, A. Gottlieb, The method of quantum clustering, in: Proceedings of Neural Information Processing Systems NIPS 2001, 2001, pp. 769–776.

[20] R. V. Casaña-Eslava, J. D. Martín-Guerrero, I. H. Jarman, P. J. G. Lisboa, Performance assessment of quantum clustering in non-spherical distributions, in: Proceedings of the 24th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2016, pp. 339–344.

[21] R. V. Casaña-Eslava, I. H. Jarman, P. J. Lisboa, J. D. Martín-Guerrero, Quantum clustering in non-spherical data distributions: Finding a suitable number of clusters, Neurocomputing 268 (2017) 127–141.

[22] C. Meek, Causal inference and causal explanation with background knowledge, in: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1995, pp. 403–410.

[23] J. Pearl, et al., Models, reasoning and inference (2000).

[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd international conference on Knowledge Discovery and Data mining (KDD'96), AAAI Press, 1996, pp. 226–231.

[25] R. Marimont, M. Shapiro, Nearest neighbour searches and the curse of dimensionality, IMA Journal of Applied Mathematics 24 (1) (1979) 59–70.

[26] E. Chávez, G. Navarro, R. Baeza-Yates, J. L. Marroquín, Searching in metric spaces, ACM computing surveys (CSUR) 33 (3) (2001) 273–321.

[27] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, P. Lamere, The Million Song Dataset., in: Ismir, Vol. 2, 2011, p. 10.

[28] I. Jolliffe, Principal component analysis, Springer Series in Statistics, Berlin: Springer, 1986 1.

[29] W. S. Torgerson, Multidimensional scaling: I. Theory and method, Psychometrika 17 (4) (1952) 401–419.

[30] G. Young, A. S. Householder, Discussion of a set of points in terms of their mutual distances, Psychometrika 3 (1) (1938) 19–22.

[31] J. C. Gower, Some distance properties of latent root and vector methods used in multivariate analysis, Biometrika 53 (3-4) (1966) 325–338.

[32] J. Nash, C1 isometric imbeddings, Annals of mathematics (1954) 383–396.

[33] J. Nash, The imbedding problem for Riemannian manifolds, Annals of mathematics (1956) 20–63.

[34] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[35] Y. Zhao, S. Tasoulis, T. Roos, Manifold Visualization via Short Walks, Eurographics Conference on Visualization (EuroVis) 2016.

[36] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[37] E. Fix, J. L. Hodges Jr, Discriminatory analysis. Nonparametric discrimination: consistency properties, Tech. rep., DTIC Document (1951).

[38] K. Q. Weinberger, L. K. Saul, Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research 10 (Feb) (2009) 207–244.

[39] J. Goldberger, G. E. Hinton, S. T. Roweis, R. Salakhutdinov, Neighbourhood components analysis, in: Advances in neural information processing systems, 2004, pp. 513–520.

[40] E. P. Xing, A. Y. Ng, M. I. Jordan, S. Russell, Distance metric learning with application to clustering with side-information, Advances in neural information processing systems 15 (2003) 505–512.

[41] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of eugenics 7 (2) (1936) 179–188.

[42] T. Hastie, R. Tibshirani, Discriminant adaptive nearest neighbor classification, IEEE transactions on pattern analysis and machine intelligence 18 (6) (1996) 607–616.

[43] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall, Learning distance functions using equivalence relations, in: ICML, Vol. 3, 2003, pp. 11–18.

[44] N. Shental, T. Hertz, D. Weinshall, M. Pavel, Adjustment learning and relevant component analysis, in: European Conference on Computer Vision, Springer, 2002, pp. 776–790.

[45] S.-i. Amari, S. Wu, Improving support vector machine classifiers by modifying kernel functions, Neural Networks 12 (6) (1999) 783–789.

[46] T. S. Jaakkola, D. Haussler, et al., Exploiting generative models in discriminative classifiers, Advances in neural information processing systems (1999) 487–493.

[47] S.-I. Amari, Differential-geometrical methods in statistics, Vol. 28, Springer Science & Business Media, 2012.

[48] C. R. Rao, Information and the accuracy attainable in the estimation of statistical parameters, in: Breakthroughs in statistics, Springer, 1992, pp. 235–247.

[49] S.-I. Amari, Natural gradient works efficiently in learning, Neural computation 10 (2) (1998) 251–276.

[50] K. M. Carter, R. Raich, W. G. Finn, A. O. H. III, FINE: Fisher Information Nonparametric Embedding, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (11) (2009) 2093–2098. doi:10.1109/TPAMI.2009.67.

[51] S. Kaski, J. Sinkkonen, Metrics that learn relevance, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Vol. 5, 2000, pp. 547–552 vol.5. doi:10.1109/IJCNN.2000.861526.

[52] S. Kaski, J. Sinkkonen, J. Peltonen, Bankruptcy analysis with self-organizing maps in learning metrics, IEEE Transactions on Neural Networks 12 (4) (2001) 936–947. doi:10.1109/72.935102.

[53] R. W. Floyd, Algorithm 97: Shortest Path, Commun. ACM 5 (6) (1962) 345–. doi:10.1145/367766.368168.
URL http://doi.acm.org/10.1145/367766.368168

[54] S. Warshall, A Theorem on Boolean Matrices, J. ACM 9 (1) (1962) 11–12. doi:10.1145/321105.321107.
URL http://doi.acm.org/10.1145/321105.321107

[55] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik 1 (1) (1959) 269–271.

[56] M. E. Newman, Detecting community structure in networks, The European Physical Journal B-Condensed Matter and Complex Systems 38 (2) (2004) 321–330.

[57] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, Journal of Statistical Mechanics: Theory and Experiment 2005 (09) (2005) P09008.

[58] S. Fortunato, Community detection in graphs, Physics reports 486 (3) (2010) 75–174.

[59] B. W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell system technical journal 49 (2) (1970) 291–307.

[60] A. Pothen, H. D. Simon, K.-P. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM journal on matrix analysis and applications 11 (3) (1990) 430–452.

[61] J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics Springer, Berlin, 2001.

[62] M. Girvan, M. E. Newman, Community structure in social and biological networks, Proceedings of the national academy of sciences 99 (12) (2002) 7821–7826.

[63] M. E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical review E 69 (2) (2004) 026113.

[64] M. E. Newman, Fast algorithm for detecting community structure in networks, Physical review E 69 (6) (2004) 066133.

[65] R. Guimera, L. A. N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (7028) (2005) 895–900.

[66] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Physical review E 72 (2) (2005) 027104.

[67] A. K. Jain, Data clustering: 50 years beyond K-means, Pattern recognition letters 31 (8) (2010) 651–666.

[68] L. Rokach, O. Maimon, Clustering methods, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 321–352.

[69] D. Horn, A. Gottlieb, The method of quantum clustering, in: Advances in neural information processing systems, 2002, pp. 769–776.

[70] P. J. G. Lisboa, T. A. Etchells, I. H. Jarman, S. J. Chambers, Finding reproducible cluster partitions for the k-means algorithm, BMC Bioinformatics 14 (Suppl. 1) (2013) S8.

[71] H. Cramer, Mathematical Methods of Statistics (PMS-9), Princeton University Press, Princeton, NJ, USA, 1999.

[72] N. Nasios, A. G. Bors, Kernel-based classification using quantum mechanics, Pattern Recognition 40 (2006) 875–889.

[73] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Rubin, Bayesian data analysis, Vol. 2, Chapman & Hall/CRC Boca Raton, FL, USA, 2014.

[74] R. Gray, Vector quantization, IEEE Assp Magazine 1 (2) (1984) 4–29.

[75] B. W. Silverman, Density estimation for statistics and data analysis, Vol. 26, CRC press, 1986.

[76] D. Comaniciu, P. Meer, Distribution free decomposition of multivariate data, Pattern analysis & applications 2 (1) (1999) 22–30.

[77] Y. Cui, J. Shi, Z. Wang, Development of quantum local potential function networks based on quantum assimilation and subspace division, IEEE transactions on neural networks and learning systems 29 (1) (2018) 63–73.

[78] L. Wasserman, Topological data analysis, Annual Review of Statistics and Its Application 5 (2018) 501–532.

[79] P. Bubenik, Statistical topological data analysis using persistence landscapes, The Journal of Machine Learning Research 16 (1) (2015) 77–102.

[80] F. Chazal, L. J. Guibas, S. Y. Oudot, P. Skraba, Persistence-based clustering in riemannian manifolds, Journal of the ACM (JACM) 60 (6) (2013) 41.

[81] N. Nasios, A. G. Bors, Finding the Number of Clusters for Nonparametric Segmentation, in: Computer Analysis of Images and Patterns, 11^{th} International Conference CAIP, LNCS 3691, 2005, pp. 213–221.

[82] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Advances in neural information processing systems, 2005, pp. 1601–1608.

[83] Y. Li, Y. Wang, Y. Wang, L. Jiao, Y. Liu, Quantum clustering using kernel entropy component analysis, Neurocomputing 202 (2016) 36–48.

[84] P. Vincent, Y. Bengio, Manifold Parzen Windows, in: Advances in Neural Information Processing Systems, 2003, pp. 849–856.

[85] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.

[86] R. Daly, Q. Shen, S. Aitken, Learning Bayesian networks: approaches and issues, The knowledge engineering review 26 (2) (2011) 99–157.

[87] D. Bacciu, T. A. Etchells, P. J. Lisboa, J. Whittaker, Efficient identification of independence networks using mutual information, Computational Statistics 28 (2) (2013) 621–646.

[88] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the royal statistical society. Series B (Methodological) (1995) 289–300.

[89] A. Fast, M. Hay, D. Jensen, Improving accuracy of constraint-based structure learning, Tech. rep., Technical report 08-48, University of Massachusetts Amherst, Computer Science Department (2008).

[90] M. Kalisch, P. Bühlmann, Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm, J. Mach. Learn. Res. 8 (2007) 613–636.

[91] M. Kalisch, P. Bühlmann, Robustification of the PC-algorithm for Directed Acyclic Graphs, Journal of Computational and Graphical Statistics 17 (4) (2008) 773–789.

[92] D. Colombo, M. H. Maathuis, Order-independent constraint-based causal structure learning., Journal of Machine Learning Research 15 (1) (2014) 3741–3782.

[93] T. Richardson, P. Spirtes, Ancestral graph Markov models, Annals of Statistics (2002) 962–1030.

[94] M. H. Maathuis, D. Colombo, M. Kalisch, P. Bühlmann, Predicting causal effects in large-scale systems from observational data, Nature Methods 7 (4) (2010) 247–248.

[95] M. H. Maathuis, M. Kalisch, P. Bühlmann, et al., Estimating high-dimensional intervention effects from observational data, The Annals of Statistics 37 (6A) (2009) 3133–3164.

[96] D. Dash, M. J. Druzdzel, A hybrid anytime algorithm for the construction of causal models from sparse data, in: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1999, pp. 142–149.

[97] MATLAB Deep Learning Toolbox, the MathWorks, Natick, MA, USA (2018b).

[98] C. M. Bishop, Pattern recognition, Machine Learning 128.

[99] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable Parallel Programming with CUDA, Queue 6 (2) (2008) 40–53. doi:10.1145/1365490.1365500.
URL http://doi.acm.org/10.1145/1365490.1365500

[100] M. Abadi, et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org (2015).
URL https://www.tensorflow.org/

[101] G. Sharma, J. Martin, MATLAB®: a language for parallel computing, International Journal of Parallel Programming 37 (1) (2009) 3–36.

[102] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[103] S. Ghemawat, H. Gobioff, S.-T. Leung, The Google File System, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 29–43. doi:10.1145/1165389.945450.
URL http://doi.acm.org/10.1145/1165389.945450

[104] S. Owen, S. Owen, Mahout in action, Manning Shelter Island, 2012.

[105] R. Fletcher, Practical methods of optimization, John Wiley & Sons, 2013.

[106] C. Zheng, J. Wang, All-Pairs-Shortest-Paths in Spark, Tech. rep., Stanford: Institute for Computational & Mathematical Engineering (2015).

[107] F. Lin, W. W. Cohen, Power Iteration Clustering., in: ICML, Vol. 10, 2010, pp. 655–662.

[108] J. L. Brisman, J. K. Song, D. W. Newell, Cerebral aneurysms, New England journal of medicine 355 (9) (2006) 928–939.

[109] M. Villa-Uriol, G. Berti, D. Hose, A. Marzo, A. Chiarini, J. Penrose, J. Pozo, J. Schmidt, P. Singh, R. Lycett, et al., @ neurIST complex information processing toolchain for the integrated management of cerebral aneurysms, Interface Focus (2011) rsfs20100033.

[110] P. Bijlenga, C. Ebeling, M. Jaegersberg, P. Summers, A. Rogers, A. Waterworth, J. Iavindrasana, J. Macho, V. M. Pereira, P. Bukovics, et al., Risk of rupture of small anterior communicating artery aneurysms is similar to posterior circulation aneurysms, Stroke 44 (11) (2013) 3018–3026.

[111] J. P. Greving, M. J. Wermer, R. D. Brown Jr, A. Morita, S. Juvela, M. Yonekura, T. Ishibashi, J. C. Torner, T. Nakayama, G. J. Rinkel, et al., Development of the PHASES score for prediction of risk of rupture of intracranial aneurysms: a pooled analysis of six prospective cohort studies, The Lancet Neurology 13 (1) (2014) 59–66.

[112] D. Backes, M. D. Vergouwen, A. T. Tiel Groenestege, A. S. E. Bor, B. K. Velthuis, J. P. Greving, A. Algra, M. J. Wermer, M. A. van Walderveen, K. G. terBrugge, et al., PHASES score for prediction of intracranial aneurysm growth, Stroke 46 (5) (2015) 1221–1226.

[113] C. Valencia, M. Villa-Uriol, J. Pozo, A. Frangi, Morphological descriptors as rupture indicators in middle cerebral artery aneurysms, in: Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, IEEE, 2010, pp. 6046–6049.

[114] D. Dheeru, E. Karra Taniskidou, "UCI Machine Learning Repository" (2017).
URL `http://archive.ics.uci.edu/ml`

[115] D. Kinga, J. B. Adam, A method for stochastic optimization, in: International Conference on Learning Representations (ICLR), Vol. 5, 2015, pp. –.

[116] K. B. Petersen, M. S. Pedersen, The Matrix Cookbook, Technical University of Denmark, 2012, version 20121115.
URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`

[117] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment 2008 (10) (2008) P10008.

[118] D. Horn, A. Gottlieb, Algorithm for data clustering in pattern recognition problems based on quantum mechanics, Physical review letters 88 (1) (2001) 018702.

[119] M. Forina, C. Armanino, S. Lanteri, E. Tiscornia, Food Research and Data Analysis, Applied Science Publishers, 1983.

[120] I. Tsamardinos, L. E. Brown, Bounding the False Discovery Rate in Local Bayesian Network Learning., in: AAAI, 2008, pp. 1100–1105.

[121] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29.

[122] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, G. F. Cooper, The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in: AIME 89, Springer, 1989, pp. 247–256.

[123] M. Julià-Sapé, D. Acosta, M. Mier, C. Arùs, D. Watson, et al., A multi-centre, web-accessible and quality control-checked database of in vivo MR spectra of brain tumour patients, Magnetic Resonance Materials in Physics, Biology and Medicine 19 (1) (2006) 22–33.

[124] A. R. Tate, J. Underwood, D. M. Acosta, M. Julià-Sapé, C. Majós, À. Moreno-Torres, F. A. Howe, M. Van Der Graaf, V. Lefournier, M. M. Murphy, et al., Development of a decision support system for diagnosis and grading of brain tumours using in vivo magnetic resonance single voxel spectra, NMR in Biomedicine 19 (4) (2006) 411–434.

[125] E. J. Delikatny, S. Chawla, D.-J. Leung, H. Poptani, MR-visible lipids and the tumor microenvironment, NMR in biomedicine 24 (6) (2011) 592–611.

[126] V. Govindaraju, K. Young, A. A. Maudsley, Proton NMR chemical shifts and coupling constants for brain metabolites, NMR in Biomedicine: An International Journal Devoted to the Development and Application of Magnetic Resonance In Vivo 13 (3) (2000) 129–153.

[127] H. Poptani, R. Gupta, V. Jain, R. Roy, R. Pandey, Cystic intracranial mass lesions: possible role of in vivo MR spectroscopy in its differential diagnosis., Magnetic resonance imaging 13 (7) (1995) 1019.

[128] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, P. Bühlmann, Causal Inference Using Graphical Models with the R Package pcalg, Journal of Statistical Software 47 (11) (2012) 1–26.
URL `http://www.jstatsoft.org/v47/i11/`

[129] H. Nakamura, M. Doi, T. Suzuki, Y. Yoshida, M. Hoshikawa, M. Uchida, Y. Tanaka, M. Takagi, Y. Nakajima, The significance of lactate and lipid peaks for predicting primary neuroepithelial tumor grade with proton MR spectroscopy, Magnetic Resonance in Medical Sciences 17 (3) (2018) 238–243.

[130] P. W. Laird, R. S. Tedlow, New and Improved: The Story of Mass Marketing in America (1991).

[131] R. S. Tedlow, G. G. Jones, The Rise and Fall of Mass Marketing (RLE Marketing), Vol. 25, Routledge, 2014.

[132] J. Hoek, P. Gendall, D. Esslemont, Market segmentation: A search for the Holy Grail?, Journal of Marketing Practice: Applied Marketing Science 2 (1) (1996) 25–34.

Publications

This part contains only the list of the works already published during the research period leading to this thesis, listed in chronological order.

1. **Performance assessment of quantum clustering in non-spherical data distributions [20].**
   Raúl V. Casaña-Eslava, José D. Martín-Guerrero, Ian H. Jarman and Paulo J. G. Lisboa
   ESANN 2016 proceedings. Bruges (Belgium)

2. **Quantum clustering in non-spherical data distributions: finding a suitable number of clusters [21].**
   Raúl V. Casaña-Eslava, Ian H. Jarman, Paulo J. G. Lisboa and José D. Martín-Guerrero
   Neurocomputing. (2017)

3. **Structure finding stabilization and optimization with the PC algorithm.**
   Raúl V. Casaña-Eslava, Ian H. Jarman, Sandra Ortega-Martorell, Paulo J. Lisboa, José D. Martín-Guerrero
   Computational Intelligence methods for Bioinformatics and Biostatistics, CIBB18.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Performance assessment of quantum clustering in non-spherical data distributions

Raúl V. Casaña-Eslava[1], José D. Martín-Guerrero[2], Ian H. Jarman[1] and Paulo J. G. Lisboa[1]

1- School of Computing and Mathematical Sciences, Liverpool John Moores University, United Kingdom
2- Department of Electronic Engineering, University of Valencia, Spain

Abstract. This work deals with the performance of Quantum Clustering (QC) when applied to non-spherically distributed data sets; in particular, QC outperforms K-Means when applied to a data set that contains information of different olive oil areas. The Jaccard score can be set depending on QC parameters; this enables to find local maxima by tuning QC parameters, thus showing up the underlying data structure. In conclusion, QC appears as a promising solution to deal with non-spherical data distributions; however, some improvements are still needed, for example, in order to find out a way to detect the appropriate number of clusters for a given data set.

1 Introduction.

K-means is the most known and widely-used clustering algorithm; however, it has a number of problems, being two of the most important ones the fact that the number of clusters is not automatically selected and its difficulty to cluster properly when the dataset is not spherically distributed. Numerous works have been carried out in order to face the former problem; for instance, [1] and [2] make use of Cramérs' V statistic as stability measure to produce the Separation Concordance (SeCo) map, and then using the Area Under this Curve as metric to obtain the most consistent values of K. Nevertheless, the latter problem is difficult to be solved because of k-means design itself. In this framework, Quantum Clustering (QC) appears as a promising solution due to its ability to work well with data non-spherically distributed data..

The QC was introduced in [3] using the Schrodinger equation on probability wave function formed as a superposition of N Gaussian probability functions (1), where there are N data points of dimension d. Then, looking for solutions of the harmonic oscillator potential in ground energy eigenstate, (2 – 4), those centroids in which the potential has a local minima can be found. From the wave function in (1) the potential function $V(x)$ obtains the σ parameter; more minima appear in $V(x)$ as σ is decreased. Tuning σ can also be used for the estimation of the appropriate number of clusters.

$$\psi(x) = \sum_i^N e^{-\frac{(x-x_i)^2}{2\sigma^2}} \qquad (1)$$

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

$$H\psi \equiv \left(-\frac{\sigma^2}{2}\nabla^2 + V(x)\right)\psi = E\psi \quad (2)$$

$$V(x) = E + \frac{\frac{\sigma^2}{2}\nabla^2\psi}{\psi} \quad (3)$$

$$E = -min\frac{\frac{\sigma^2}{2}\nabla^2\psi}{\psi} \quad (4)$$

QC has already been tested in [4]; this work made use of Single Value Decomposition (SVD) as a preprocessing step before the QC algorithm. Three known datasets of cells and genes were tested obtaining good results when dimensions were truncated to the 4th – 5th principal components before the application of QC; the corresponding Jaccard scores outperformed those achieved by k-means.

The interface called Comparative-Package-for-Clustering-Assessment (COMPACT) [5] was used to obtain the results shown in this paper. COMPACT implements several clustering algorithms and has the option of reducing the dataset´s dimensionality using SVD. The Jaccard score is used to evaluate the clusters obtained compared with the known outputs.

2 The olive oil data set

The known olive oil dataset [6] has been chosen because it presents a non-spherical distribution, and hence, it is suitable to evaluate QC performance compared to k-means. The data set presents two types of underlying structure (3 regions and 9 sub-regions) thus making the choice of the number of clusters challenging.

The olive oil dataset, consists of 572 observations with 8 characteristics, related to the fatty acid content of olive oil. This data corresponds to 3 collection regions, and 9 sub-regions.; four from Southern Italy (North and South Apulia, Calabria and Sicily), three from Umbria (Umbria, East and West Liguria) and two from Sardinia (Inland and Coastal regions).
The projected visualization of the underlying dataset is shown in Figure 1, where each data point is labelled according to the region from which it was obtained [7]; the overlapping of the data from Calabria, North and South Apulia and Sicily is remarkable.

3 QC setting-up

As previously mentioned, COMPACT was used as interface to evaluate the QC performance. There are some parameters related to preprocessing (SVD, normalization, component reduction, etc.) … and others directly related to the QC algorithm; among the latter, the most important parameter is σ, although there are more algorithm

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

parameters that can be tuned (number of steps, rescale, QC core, % pure terms and learning rate η).

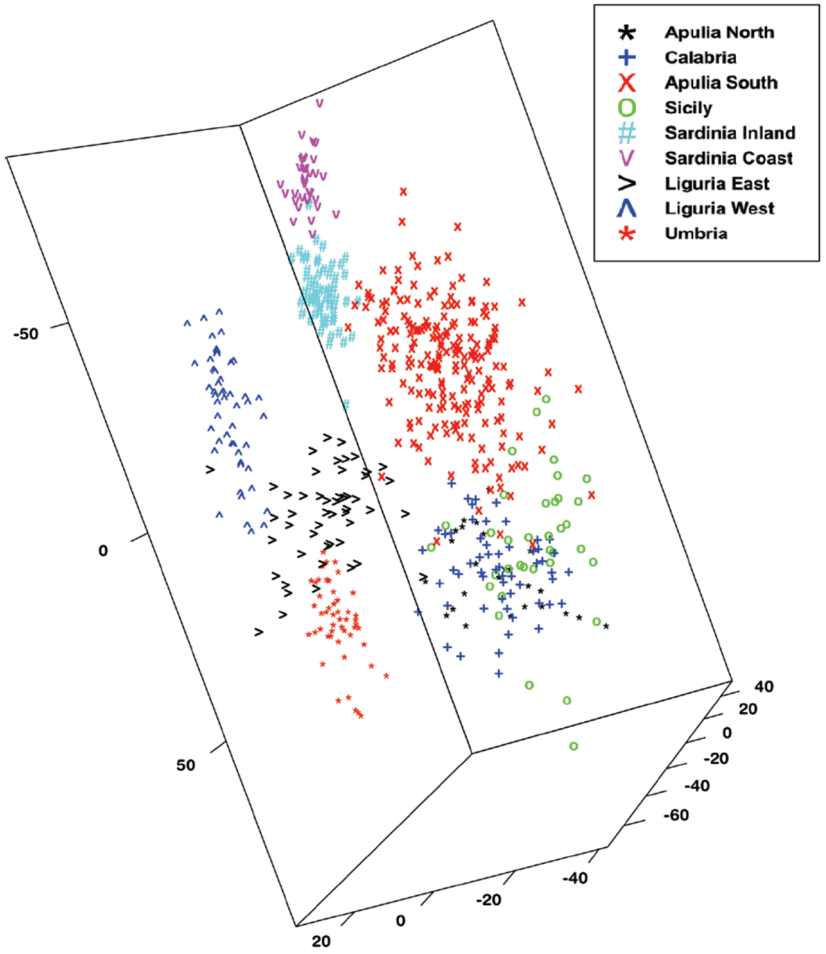


Fig. 1: Visualization of the 3 main principal components of olive oil data.

An evaluation performance based on the Jaccard score allowed to draw a number of conclusions:

- Normalization and SVD is needed in order to avoid a high number of clusters.
- When QC core is applied, the option of % of pure elements just tends to remove observations with an outlier behavior (the performance changes because the number of observations decreases).
- The optimal value of the learning rate is $\eta = 0.1$; other values of η might improve the performance but involving an unstable range of σ to assign a reasonable number of clusters.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Finally, the best performance was obtained with the following combination of parameters:

- SVD pre-processing is enabled
- Normalization is applied
- QC core is not activated
- The σ belongs to [0.4, 0.6] producing 10 to 2 clusters.
- The η is 0.1
- Number of steps: 100

4 Data structure

Since the actual output of the olive oil data set is known (classes and sub-classes), it is possible to assess and analyze the performance of QC. Although one the main advantages of the QC is that the underlying data structure can be found by varying the parameter σ. Figure 2 shows that QC does not find the correct number of clusters when it obtains the best performance results, that are highlighted in black; in particular the value of σ that provides the best performance in the case of three-cluster problem leads to four clusters, while QC finds eight clusters when it achieves the best performance in the nine-cluster problem.

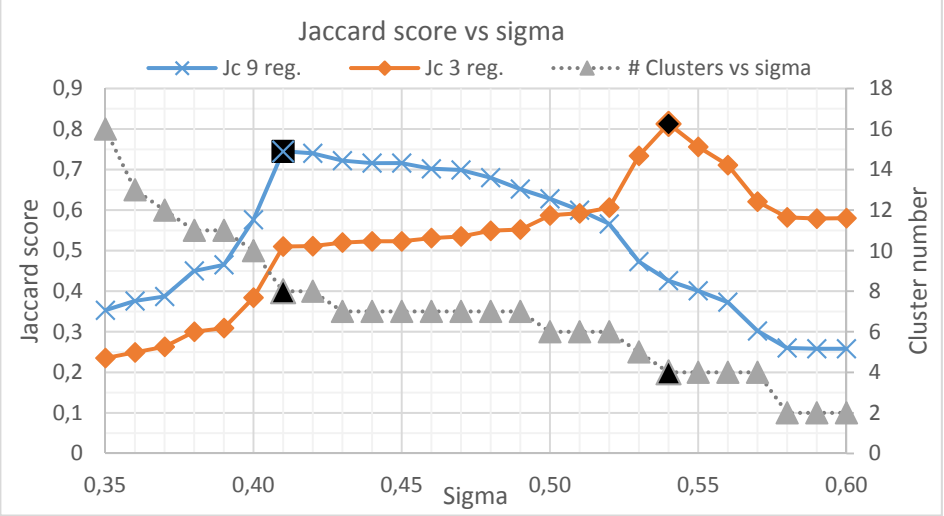


Fig. 2: Performance as σ function. Left axis shows Jaccard score and right axis shows the cluster number. Best Jaccard score results alongside the number of clusters are highlighted in black.

5 QC *vs* K-means performance

This section benchmarks QC performance versus K-Means, for the two classifications of the data set, namely, three and nine clusters. It must be emphasized that an additional

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

advantage of QC with respect to K-Means is that QC obtains the same solution every time the algorithm is run for a particular σ value whereas K-means may provide different solutions in different runs since it is strongly depends on the initial conditions; to circumvent that bias, K-means was run 500 times to estimate the lowest SSE for each cluster number, [2, 12], then the Jaccard score has been obtained for the 3 and 9 regions, bearing in mind the lowest SSE doesn't imply the highest Jaccard score.

The results presented in Fig. 3 show that although QC nor K-Means find the correct number of clusters in either of the two problems, QC performance is considerably better than that achieved by K-Means, even taking into account that the right number of clusters is provided to K-Means. The best solutions for 3 regions are 4 clusters in both cases, with the QC Jaccard higher than K-Means. The best matching results for 9 sub-regions with K-Means is 6 clusters and QC 8 clusters, again QC scoring slightly higher (0.74 vs 0.72). The K-Means SSE decreases as K increases, as expected.

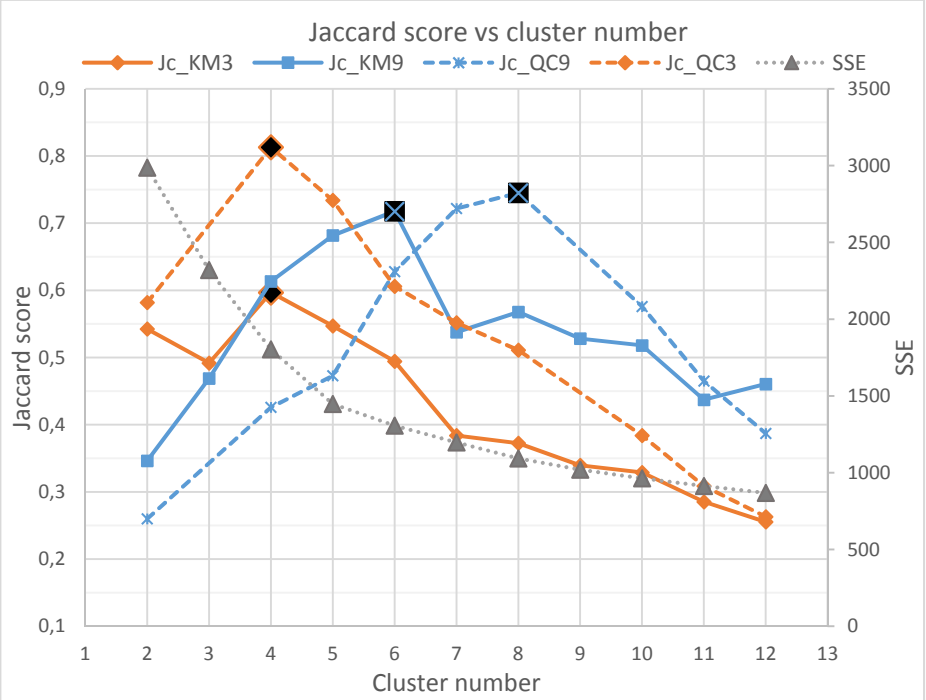


Fig. 3 Jaccard score of K-means and QC. Left axis represents Jaccard score and right axis represents SSE of K-Means. Orange lines refer to 3 cluster problem and blue ones to 9 cluster problem. Dashed line shows QC performance and the grey dot line refers to SSE of K-Means.

Table 1 depicts the best indices for both algorithms: Jaccard score, purity and efficiency (η).

Best Jaccard sc.	K-Means				QC			
Regions	Clust.	Jaccard	Purity	η	Clust.	Jaccard	Purity	η
3	4	0,597	0,619	0,943	4	0.813	0.905	0.889
9	6	0,718	0,911	0,772	8	0.745	0.794	0.924

Table 1: Jaccard score, purity and efficiency (η) are shown of clusters with the best Jaccard scores.

6 Conclusion

This work has proposed the use of QC to cluster non-spherical data distributions. QC outperforms the classical K-Means when applied to a data set containing information of different production regions of olive oil. Although, QC may not find the correct number of clusters, the performance measured in terms of Jaccard score, purity and efficiency is much better than that achieved by a K-Means that does know the number of clusters in advance.
Our ongoing and future research is related to the application of QC in more demanding environments in order to figure out its usefulness and range of application. And also, research related to the search for the correct σ for unsupervised data.

Acknowledgements:
This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under project with reference number TIN2014-52033-R.

References

[1] P. J. G. Lisboa, T. A. Etchells, I. H. Jarman y S. J. Chambers, «Finding reproducible cluster partitions for the k-means algorithm,» *BMC Bioinformatics,* vol. 14, nº Suppl. 1, p. S8, 2013.

[2] S. J. Chambers, I. H. Jarman, T. A. Etchells and P. J. G. Lisboa, "Inference of number of prototypes with a framework approach to K-means clustering," *Int. J. Biomedical Engineering and Technology,* vol. 13, no. 4, pp. 323-340, 2013.

[3] D. Horn y A. Gottlieb, «Algorithm for Data Clustering in Pattern Recognition Problems Based on Quantum Mechanics,» *Physical Review Letters,* vol. 88, nº 1, p. 018702, 2002.

[4] D. Horn y I. Axel, «Novel clustering algorithm for microarray expression data in a truncated SVD space,» *Bioinformatics,* vol. 19, nº 9, pp. 1110-5, 2003.

[5] R. Varshavsky, M. Linial y D. Horn, COMPACT: A Comparative Package for Clustering Assessment, SpringerVerlag, 2005, pp. 159-167.

[6] M. Forina, C. Armanino, S. Lanteri y E. Tiscornia, «Classification of olive oils from their fatty acid composition,» de *Food Research and data Analysis, Applied Science Publishers*, London, In: Martens, M., Russwurm, H. Jr., 1983, pp. 189-214.

[7] S. J. Chambers, «A framework approach to initialisation dependent clustering.,» PhD Thesis, Liverpool John Moores University, 2015.

[8] N. Nasios y A. G. Bors, «Finding the Number of Clusters for Nonparametric Segmentation,» de *Computer Analysis of Images and Patterns*, Springer Berlin Heidelberg, 2005, pp. 213-221.

Neurocomputing 268 (2017) 127–141

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Quantum clustering in non-spherical data distributions: Finding a suitable number of clusters

Raúl V Casaña-Eslava[a,*], Ian H. Jarman[a], Paulo J.G. Lisboa[a], José D. Martín-Guerrero[b]

[a] Department of Applied Mathematics. Liverpool John Moores University. James Parsons Building, 3 Byrom Street. L3 3AF Liverpool (Merseyside), United Kingdom

[b] Department of Electronic Engineering, University of Valencia. ETSE-UV, Avgda. Universitat, s/n. E46100 Burjassot (Valencia) - Spain

ARTICLE INFO

Article history:
Received 8 July 2016
Revised 5 December 2016
Accepted 19 January 2017
Available online 29 April 2017

Keywords:
Quantum clustering
Non-spherical data distributions
Number of clusters
Parameter optimisation
Separation and Concordance

ABSTRACT

Quantum Clustering (QC) provides an alternative approach to clustering algorithms, several of which are based on geometric relationships between data points. Instead, QC makes use of quantum mechanics concepts to find structures (clusters) in data sets by finding the minima of a quantum potential. The starting point of QC is a Parzen estimator with a fixed length scale, which significantly affects the final cluster allocation. This dependence on an adjustable parameter is common to other methods. We propose a framework to find suitable values of the length parameter σ by optimising twin measures of cluster separation and consistency for a given cluster number. This is an extension of the Separation and Concordance framework previously introduced for K-means clustering. Experimental results on two synthetic data sets and three challenging real-world data sets show that optimisation of cluster separation identifies QC solutions with consistently high Jaccard score measured against true-cluster labels while optimisation of cluster consistency provides insights into hierarchical cluster structure.

1. Introduction

As interest in knowledge extraction from data grows, this typically includes exploratory analysis especially when the data are unlabelled. A central step in exploratory data analysis is the discovery of different categories or profiles in the data. Clustering algorithms are efficient methods for unsupervised learning among which a frequently used algorithm is K-Means [1]. This method implements a hard partition of the data by identifying representative points, the prototypes, which minimise the sum of within cluster squared Euclidean distances as shown in Eqs. (1) and (2):

$$J(\Theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij} \cdot d(x_i, \Theta_j) \tag{1}$$

$$u_{ij} = \begin{Bmatrix} 1, & d(x_i, \Theta_j) = \min_{k=1,\ldots,K} d(x_i, \Theta_k) \\ 0, & otherwise \end{Bmatrix} \quad i = 1, \ldots, N \tag{2}$$

where $d(x_i, \Theta_j)$ is the distance between the i-th pattern and the j-th prototype, N the number of patterns and K the number of clusters. In spite of its simplicity, K-Means is an adequate and efficient choice of clustering algorithm in many cases. However, it suffers from a number of drawbacks that limits its applicability. In particular, it tends to find spherical clusters formed by approximately the same number of patterns. Moreover, the final cluster allocation vary significantly with the choice of prototype initialisation. In addition, there is a requirement to pre-set the number of clusters, K, even though the optimal value of K is generally not known in advance. Consequently, K-means may mix natural clusters or break them up with unnecessary intermediate clusters [2–4]. A previous publication [3] proposed a framework to ensure that optimal results can be reproduced when K-means is repeatedly applied to the same data. This framework relies on a parametrisation of the set of clustering solutions obtained for different prototype initialisations and cluster numbers, using measurements of cluster separation and of the internal consistency, or concordance, between multiple clustering solutions obtained for the same K, hence the term SeCo for Separation and Concordance mapping of the space of clustering solutions.

This paper proposes an extension of this method to find suitable length parameters when applying Quantum Clustering (QC) [5–7]. This alternative clustering methodology is attractive because it more naturally fits non-spherical data distributions [8,9] and it is also better suited to model clusters of different

* Corresponding author.
E-mail addresses: r.v.casanaeslava@ljmu.ac.uk, raulcasana@gmail.com (R.V. Casaña-Eslava), jose.d.martin@uv.es (J.D. Martín-Guerrero).

http://dx.doi.org/10.1016/j.neucom.2017.01.102
0925-2312/

sizes present in the same data set. We start with a review of existing methods for optimisation of the value of the scale parameter directly from the dispersion properties of the data, initially proposed in [10,11], before comparing the results with the proposed alternative method for estimating the length parameter σ using the outcome of QC clustering rather than the data alone.

Work in [12] proposes a combined methodology using kernel entropy pre-processing followed by quantum clustering. This two length scales, namely a sigma for the entropy kernel and one for quantum clustering. The experimental results reported in the paper assume that the two values are the same and estimates this as the small sample covariance for a local neighbourhood with a fixed sample size of 50. While this may be appropriate when the number of data points is large as is all of the synthetic cases reported, it remains a parameter to be adjusted. In particular, it is found in [12] that the clustering is sub-optimal for the Wisconsin Breast Cancer data set. It remains the case, therefore, that the value of sigma needs to be adjusted.

To tackle the problem of high-dimensional data and their scalability, QC can be combined with techniques of dimensionality reduction [12–15].

The rest of the paper is outlined as follows. Section 2 introduces the QC algorithm. Section 4, reviews methods for estimating optimal values of σ from the data, by application to a set of benchmarking data sets which include two synthetic examples and three real-world data sets. This is followed in Section 5, by the introduction of the SeCo framework and description of its application to the same data sets to set the length scale from clustering results. The experimental results are discussed in Section 6 from which conclusions are then drawn in Section 7.

2. Methodology

2.1. Quantum clustering

Many clustering algorithms are based on locations of points in the data space. That methodology works sufficiently well in many situations but it describes a problem that might be ill defined. QC proposes a different methodology, inspired in concepts from Quantum Mechanics [5–7]. It starts with a Parzen-window estimator of the probability distribution based on the data; then, a Gaussian kernel generates a probability distribution from the data points in a Euclidean space, as shown in Eq. (3):

$$\Psi(\mathbf{x}) = \sum_i \exp\left(-\frac{(\mathbf{x}-\mathbf{x_i})^2}{2\cdot\sigma^2}\right) \tag{3}$$

where $\mathbf{x_i}$ are the data points. QC associates maxima of this function with cluster centres in a Hilbert space driven by the Schrödinger equation so that minima of the Schrödinger potential are associated with cluster prototypes. The Schrödinger equation is given by Eq. (4):

$$H\Psi \equiv \left(-\frac{\sigma^2}{2}\nabla^2 + V(\mathbf{x})\right)\Psi(\mathbf{x}) = E\Psi(\mathbf{x}) \tag{4}$$

where $\Psi(\mathbf{x})$ is a solution of the equation (eigenstate), H is the Hamiltonian, V the potential energy and E is an energy eigenvalue. The simplest case is given by a single Gaussian where Ψ represents a single point at x_1. It leads to the potential $V = \frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x_1})^2$; this is a well-known potential in Quantum Mechanics, the so-called harmonic potential whose ground state corresponds to the eigenvalue $E = \frac{\hbar\omega}{2} = \frac{d}{2}$, where $\hbar$ is the reduced Planck constant, ω the angular frequency, and d the space dimension. Therefore, the Gaussian function describes the ground state of H.

Although in Quantum Mechanics the usual strategy is to find solutions for $\Psi(\mathbf{x})$ given the potential, the proposal of QC is the other way around, i.e., since $\Psi(\mathbf{x})$ is already determined by the data points, the goal is to find a potential $V(\mathbf{x})$ whose solution is the given $\Psi(\mathbf{x})$:

$$\begin{aligned} V(\mathbf{x}) &= E + \frac{\frac{\sigma^2}{2}\nabla^2\Psi}{\Psi} \\ &= E - \frac{d}{2} + \frac{1}{2\sigma^2\Psi}\sum_i (\mathbf{x}-\mathbf{x_i})^2 \exp\left(-\frac{(\mathbf{x}-\mathbf{x_i})^2}{2\cdot\sigma^2}\right) \end{aligned} \tag{5}$$

If V is positive definite, $\min V = 0$, and hence $E = -\min\frac{\frac{\sigma}{2}\nabla^2\Psi}{\Psi}$, which implies that $0 < E < \frac{d}{2}$.

After cluster prototypes are found, the final task is to assign each pattern to a given cluster. This can be done by means of a gradient descent algorithm; defining $\mathbf{y_i}(0) = \mathbf{x_i}$, the trajectories of this point over time, $\mathbf{y_i}(t)$, is iterated as follows, where η is the learning rate that controls the speed of approaching the nearest minimum:

$$\mathbf{y_i}(t+\Delta t) = \mathbf{y_i}(t) - \eta(t)\nabla V(\mathbf{y_i}(t)) \tag{6}$$

letting $\mathbf{y_i}$ reach an asymptotic fixed value coinciding with a cluster prototype [7].

2.2. Parameter optimisation

The QC code used in this work is based on the Matlab COMPACT GUI [16]. Among the different parameters that appear in this implementation, the most important one in the QC algorithm is σ; the rest of the parameters have been set to default because that was the setup that provided the best results in [8], the original work from which this paper is an extended version:

- Learning rate, $\eta = 0.10$
- Number of steps = 100
- Rescale each step = FALSE
- Use of QC Core = FALSE

2.3. Number of clusters

As cluster prototypes are associated with potential minima in QC, and the only undetermined parameter is σ, different clustering solutions will be obtained for different values of σ. In particular, as σ is decreased, more and deeper minima are expected to be found. The tuning of σ is usually carried out by means of varying it smoothly and looking for stability of cluster solutions [7]. Our conjecture is that if one could optimise the value of σ, QC would become an automatic clustering algorithm, able to find the best combination of structures (in principle, of different shapes) that define the data.

In the next section we introduce several data sets, real and synthetic, which will be used to illustrate the application of the proposed methodology.

3. Description of the data

Five different data sets are used in this study to illustrate the application of the proposed methods: two synthetic data sets and three real data sets commonly used to benchmark clustering methods. The data sets are described in detail in Sections 3.1 and 3.2. Both synthetic data sets are generated using Gaussian distributions, some of them highly overlapped, thus producing unique structures formed by different Gaussian distributions that are difficult to separate by clustering. The real-world datasets demonstrate markedly different cluster shapes and mix clusters of different sizes in the same data.

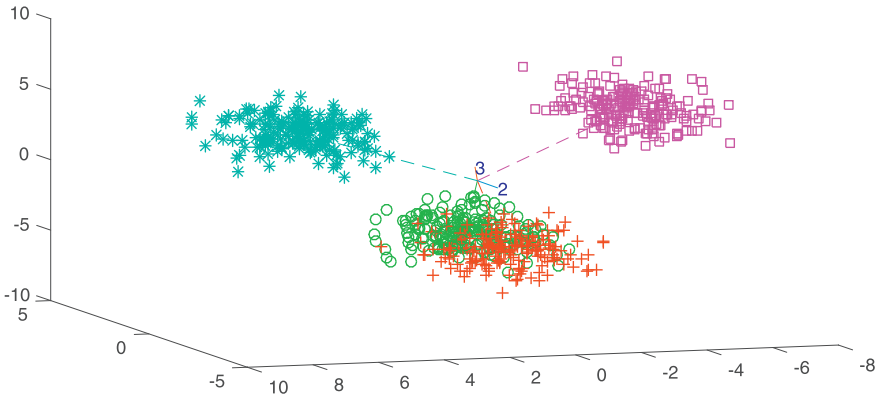


Fig. 1. Principal components' visualisation of artificial data set #1. This data set contains four clusters, generated by a Normal distribution.

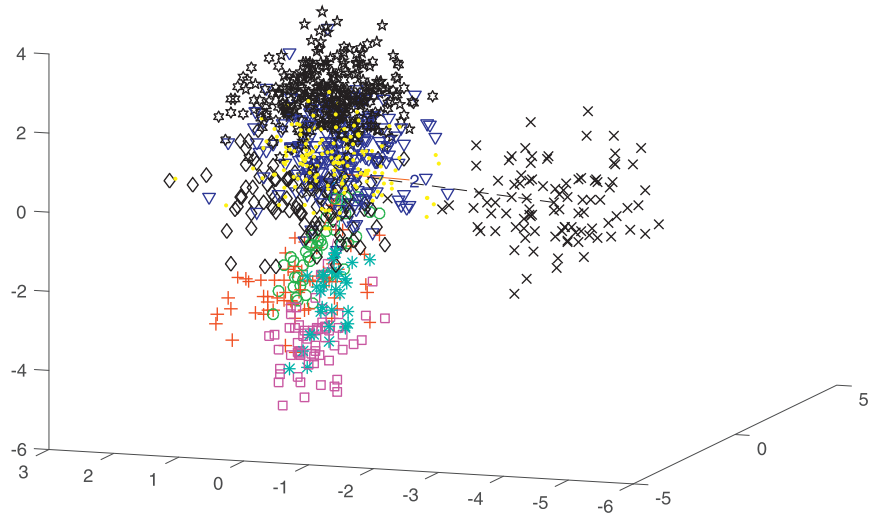


Fig. 2. Principal components' visualisation of artificial data set #2 (10 clusters). This data set contains 10 clusters, generated by a Normal distribution.

3.1. Synthetic data sets

Artificial Data Set #1 (4 clusters). This data set depicts a first possible scenario, relatively simple, formed by 800 samples in a three-dimensional space and four clusters with the same number of observations each. The aim is to evaluate how QC reacts when there are three groups of clusters equidistant and how it affects the internal Concordance when QC tries to allocate the labels with the wrong cluster number. The four clusters are generated by a spherical Normal distribution. Fig. 1 shows that two clusters are partially overlapped and the other two are totally separated.

Artificial Data Set #2 (10 clusters). This data set has been used in [2–4], it is based on 1076 observations in three dimensions with 10 clusters. Each cluster has a different proportion of observations, being some of them sparse. The overlapping between two clusters is important thus being quite difficult to detect, however there are other clusters easily separable. The covariance matrix of the Gaussian distributions is not spherical. Fig. 2 shows the principal components of this dataset.

3.2. Real data sets

Wine data set. This dataset available on the UCI data repository [17] is well known and comprises 178 observations in 13 variables. It was acquired from a chemical analysis of wines grown in one region of Italy. Each of the attributes consists of measurements taken from the various wines, which are created using three distinct cultivars. The attributes are Alcohol, Malic Acid, Ash, Alcalinity of the Ash, Magnesium, Total Phenols, Flavanoids, Nonflavanoid Phenols, Proanthocyanins, Colour Intensity, Hue, OD280/OD315 of diluted wines and Proline. The cultivars are well separated with the expectation of good classification by approaches like K-means.

Olive oil data set. The *Italian olive oil* data set consists of 572 samples and 10 variables. Eight variables describe the percentage composition of fatty acids found in the lipid fraction of these oils, which is used to determine their authenticity. The remaining two variables contain information about the classes, which are of two kinds: three "super-classes" at country level: North, South, and the island of Sardinia; *and* nine collection area classes: three from the

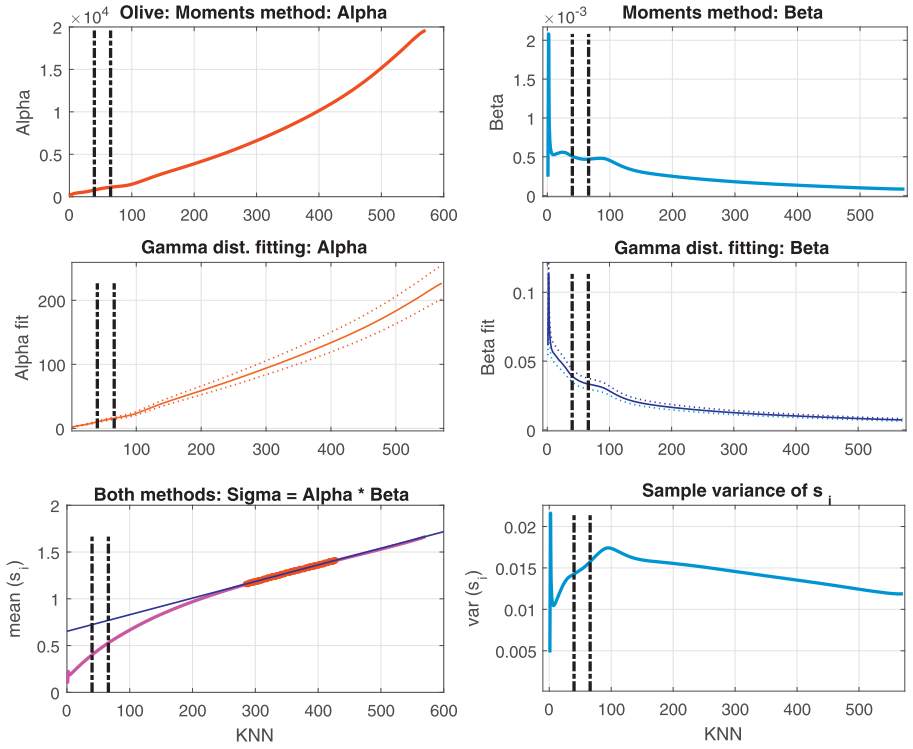


Fig. 3. Olive oil data set: two methods to estimate $\hat{\sigma}$. Moments method in top graphs, fitting Gamma distribution in middle graphs. Estimated $\hat{\sigma}$ in bottom left and s_i sample standard deviation in bottom right. Vertical lines indicate the best σ solution according to the Jaccard score for the two external labels of the olive oil data set.

Northern region (Umbria, East and West Liguria), four from the South (North and South Apulia, Calabria, and Sicily), and two from the island of Sardinia (inland and coastal Sardinia).

The goal is to distinguish the oils from different regions and areas in Italy based on their combinations of the fatty acids. The clusters corresponding to classes all have different shapes in the eight-dimensional data space defined by the concentration of fatty acids [18,19].

Iris data set. The Iris dataset [17] was introduced by Sir Ronald Fisher in 1936 for the purpose of using it as an example in explaining discriminant analysis. The dataset comprises 150 data points in four dimensions matching the Sepal and Petal width and height for each observation. There are three cohorts present in the data: Setosa, Virginica and Versicolor.

3.3. Data pre-processing

The QC algorithm is designed to work in a normalised data space so that σ values are bounded in the range [0, 2] [5–7,16]. For that reason, it is necessary to implement a previous data preprocessing.

The first step is to apply the reduced Single Value Decomposition, using the $U_{m\text{x}n}$ matrix of left-singular vectors as the new data.

The data need to be normalised to a unit hyper-sphere. However, in order to preserve length information, an extended vector is used with a column of ones added to the original matrix, $U_{m\text{x}n}$. In addition, the original data matrix is re-scaled by a single factor λ to ensure that mean length of the rows is 1. In summary:

$$Data_{m\text{x}n} = U_{m\text{x}n}\Sigma_{n\text{x}n}V^*_{n\text{x}n}$$

$$U'_{m\text{x}n} = U_{m\text{x}n}/\lambda \tag{7}$$
$$Z = rnorm([U', 1]_{m\text{x}(n+1)})$$

where *rnorm* is a function that normalises every matrix row by length 1.

In this way, raw data is transformed in a normalised hyper-sphere space, but keeping sample module information, and where the variance s_i is bounded to [0, 2].

In some datasets QC performance can be improved reducing the data dimensionality, but in this work the option of reducing the dimensionality through PCA has been skipped so that all datasets have the same preprocessing.

4. Setting the length scale from the data

For the optimisation of σ, we make use of a statistical approach for estimating the scale parameter of a potential function presented in [10,11], that can be translated for the estimation of σ in QC. The estimation is based on calculating the average Euclidean distance to a set of neighbours for each data sample; the resulting local variances are modelled as a Gamma distribution and the scale parameter is estimated as the mean of this Gamma distribution. Given a data sample $\mathbf{x_i}$, a ranking of all other data samples according to their squared Euclidean distance to $\mathbf{x_i}$ is performed:

$$R_K(\mathbf{x_i}) = \left\{x_{(k)} \mid \|x_{(k-1)} - x_i\|^2 < \|x_{(k)} - x_i\|^2\right\} \tag{8}$$

for $k = 1, 2, \ldots, K$, where $x_{(k)}$ represents the K-nearest neighbours of $\mathbf{x_i}$, and $\|\ \|$ denotes the Euclidean distance between a data sample and x_i. Since the variance s_i of the local neighbourhood around each sample can be calculated, an empirical distribution of local

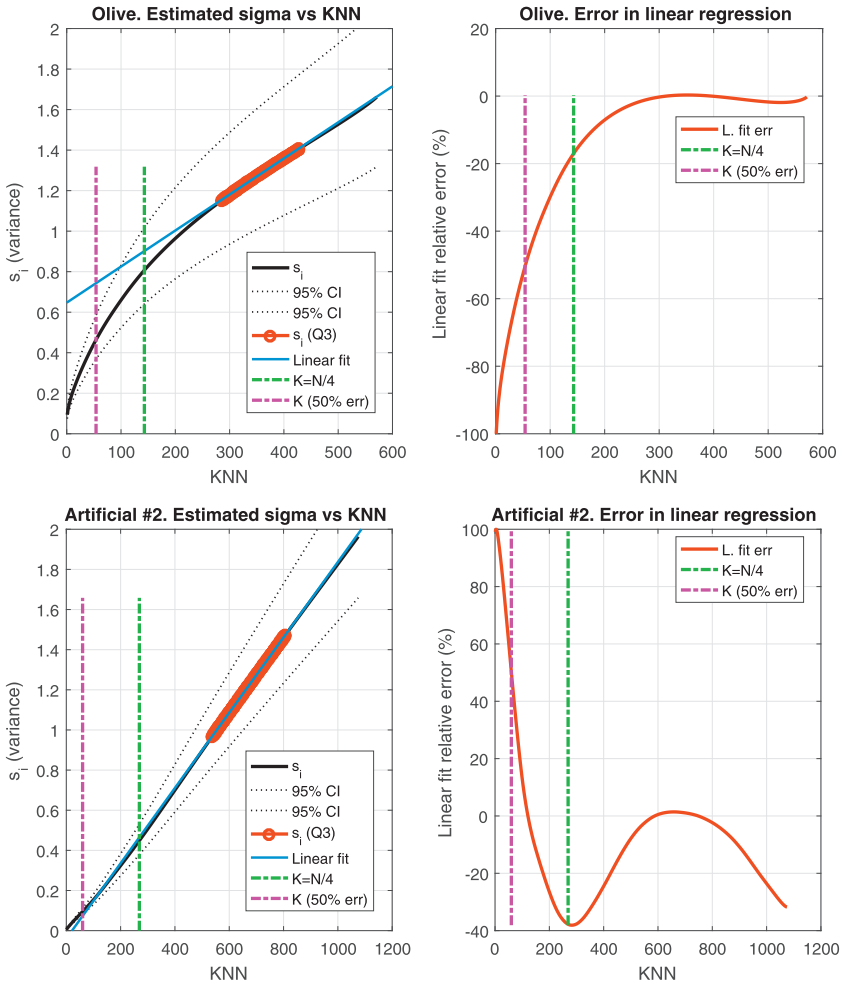


Fig. 4. Olive oil and artificial data set #2 (10 clusters): left graphs show the estimated s_i variance curve with confidence intervals at 95%, the linear regression on the interquartile range of KNN labelled $s_i(Q3)$ in red, and the two suggested KNN solutions, $K = N/4$ and K at 50% of maximum difference with respect to the linear regression. Right graphs show the error (difference) between s_i and the linear fit, and the two suggested KNN solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

variance estimates can be formed by considering several data samples $\mathbf{x_i}$ and their neighbourhoods $R_K(\mathbf{x_i})$. The probability density function that characterises the empirical local variance is modelled by the Gamma distribution:

$$p(s) = \frac{s^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha)} \exp\left(-\frac{s}{\beta}\right) \tag{9}$$

where $\alpha > 0$ is the shape parameter, and $\beta > 0$ is the scale parameter of the Gamma distribution $\Gamma(\cdot)$:

$$\Gamma(t) = \int_0^{\infty} r^{t-1} \exp(-r) dr \tag{10}$$

The parameters α and β are estimated from the empirical distribution of the variance, modelled by Eq. (9). There are different methods to calculate the parameters α and β; the moments method is proposed in [10,11]:

$$\hat{\alpha} = \left(\frac{\bar{s}}{l}\right)^2 \quad ; \quad \hat{\beta} = \frac{l^2}{\bar{s}} \tag{11}$$

where $\bar{s}$ and l are the sample mean and standard deviation of the distribution of nearest neighbour distances for a given value of K. The estimation of $\hat{\sigma}$ can then be obtained as $\hat{\sigma} = \hat{\alpha}\hat{\beta}$.

As detailed in next sections of the paper, this methodology is tested in QC to find out whether it can be successfully applied to detect a suitable number of clusters in several data sets (both synthetic and real) with different characteristics.

Two methods were used to find the most suitable fit from the data, both following the procedure described in [10,11]. The first method estimates σ using the average dispersion of the data and the second fits the distribution of the dispersion using gamma functions. We show that both methods lead to similar values of the scale parameter σ for each data set, but these values are not necessarily optimal.

The data dispersion at each data point is estimated using K-nearest neighbours (K-NN) with increasingly large numbers of near neighbours. Given a certain K, the α and β parameters of the Gamma distribution can be obtained either using the moments method as described in Eq. (11) or fitting the s_i empirical distri-

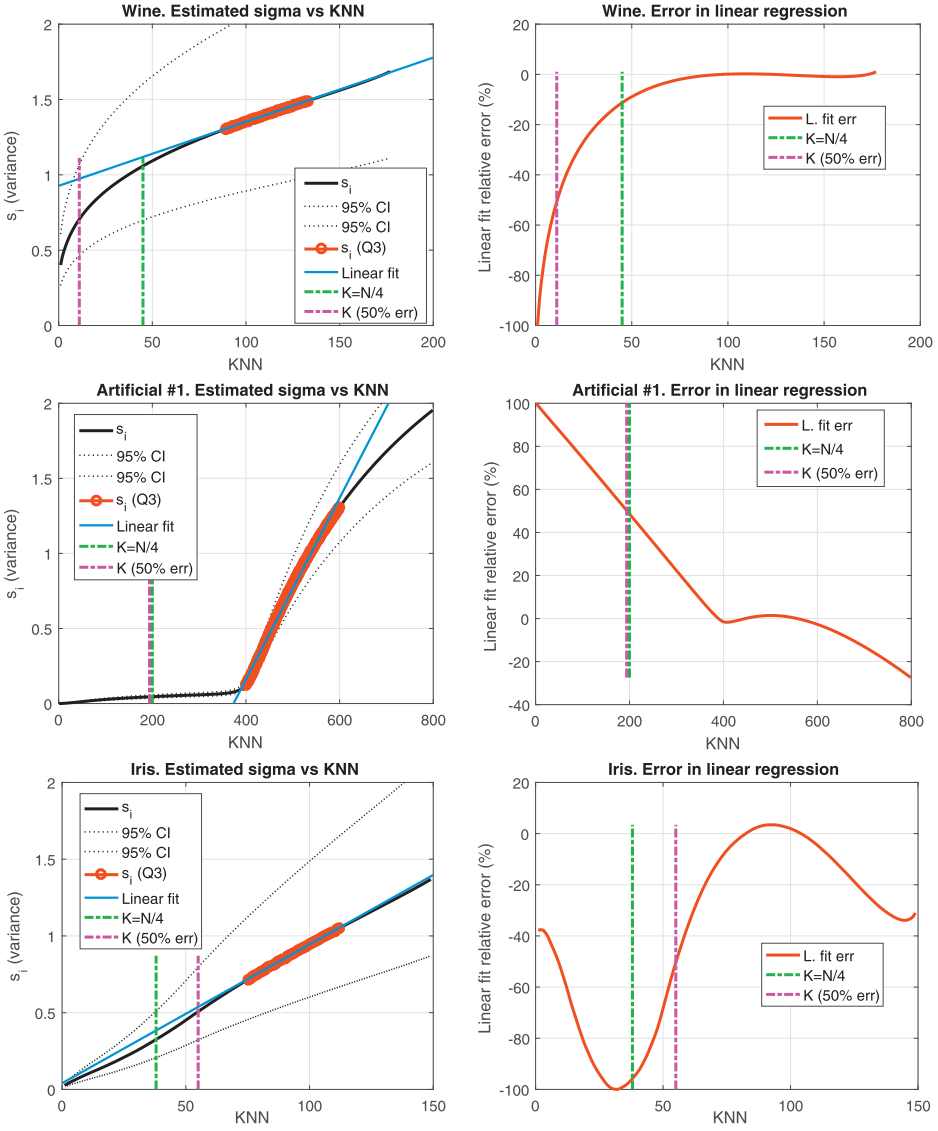


Fig. 5. Wine, artificial data set #1 (4 clusters) and Iris data set: left graphs show the estimated s_i variance curve with confidence intervals at 95%, the linear regression on the interquartile range of KNN labelled $s_i(Q3)$ in red, and the two suggested KNN solutions, $K = N/4$ and K at 50% of maximum difference w.r.t. the linear regression. Right graphs show the error between s_i and the linear fit, and the two suggested KNN solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

bution to the Gamma distribution; in this work the *gamfit* Matlab built-in function was used. Fig. 3 shows the estimation carried out by these two different ways of calculation; as expected, both produce the same result: $\hat{\sigma} = \hat{\alpha}\hat{\beta}$. The vertical lines show the best σ solution according to the Jaccard score for the two external labels of the olive oil data set; that helps to visualise the value of the optimal σ. The bottom-left graph of Fig. 3 shows the function $\sigma = f(KNN)$. Also one may observe the linear regression fitted to the interquartile range of σ values. Over $K = N/2$ a Normal distribution behaviour is expected, where the variance increases linearly as K increases. The bottom-right graph shows the s_i standard deviation with the aim of providing additional information to estimate the best K.

The next step is to decide which K is the appropriate to select σ. Two options have been discussed:

- According to [10,11], $K = N/4$, being N the sample size, is a reasonable choice. This approach suggests that the first quartile is the Separation border between the variance of close neighbours and the variance produced by remote-enough samples for Normal behaviour.
- The other option goes beyond [10,11] and it is based on the assuming that the variance has a normal behaviour when K is sufficiently large to include remote neighbours, like K in the third quartile of the sample size. Fitting a linear regression in this range of K enables to compare the variance with

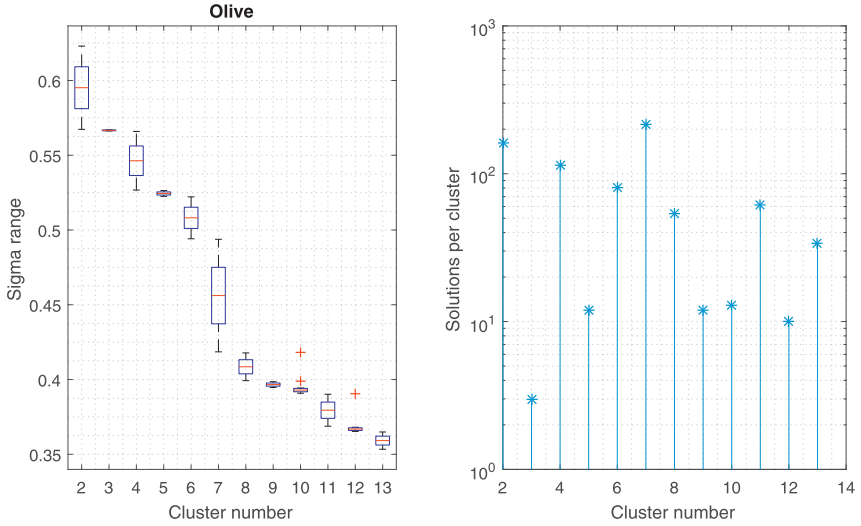


Fig. 6. Olive oil data set: left graph shows the σ range per number of clusters. Right graph shows the number of QC solutions per number of clusters. Solutions with clusters from 2 to 13 were filtered from the initial 1000 different σ values.

near neighbours against the far ones. One criterion could be to choose a K that separates more than 50% of the total distance between the variance s_i and the linear regression.

Figs. 4 and 5 present the two methods for estimating σ as a function of KNN in different datasets. One may observe a completely different s_i behaviour depending on the dataset, this affects the confidence intervals, the K selected and hence the σ estimated. This issue will be discussed thoroughly later.

It would be expectable that the s_i curves could reveal some information about the data internal structure, and that it would relate the KNN with the proper σ. But it is not the case, choosing a σ with this method seems somewhat arbitrary. There is a case to be especially noticed in Fig. 5 for the artificial data set #1, which is formed by 4 clusters, 2 of them totally separated, having 200 samples per cluster approximately; the variance curve presents an abrupt behaviour when KNN has to include observations from the more distant clusters. This should provide a clear KNN to choose σ, but the best actual σ range is about [0.65, 0.70], quite far away from the suggested [0.04, 0.05] by $K = N/4$ and $K(50\%err)$ in the middle left plot of Fig. 5. An additional problem is that this method offers a single solution that it varies strongly depending on a single premise, and hence, it is hard to create a general criterion that fits all the datasets. As QC needs more σ precision than that yielded by this method, we came up with an alternative approach, presented in Section 5.

These results illustrate the difficulty in establishing a criterion for estimation of consistently good values of the length parameter σ. This is addressed further in the next section.

5. Setting the length scale from clustering results

One of the main objectives in optimising QC to a given data set is to assess the QC solutions in an unsupervised way. This amounts to finding values of the length scale for the initial Parzen estimator, which is controlled by the Gaussian with parameter σ. In this section we will propose a framework using complementary measures of cluster performance to a) map the QC solution space, b) find suitable values for the number of clusters, K, and length parameter σ and c) generate insight into possible hierarchical structure in the data.

In the absence of external labels, we propose the use of a two-dimensional performance assessment framework, which we call Separation and Concordance (SeCo). This was first introduced in [2–4] to assess K-Means and Adaptive Resonance Theory (ART) models.

The SeCo framework characterises the quality of clustering solutions by measuring two quantities namely the cluster separation and cluster stability. The separation measure is the standard within-cluster sum of squares (SSQ) and the stability measure is the concordance between different cluster allocations for a given cluster number, quantified using the CramÃ©r V-index of association (Cv) which is a normalised version of the standard chi-squared test for contingency tables.

In order to obtain sufficient data to measure concordance reliably and also to avoid using poor clustering solutions, for each cluster number, K, a significant number of clustering solutions is obtained by varying the initial conditions which in the case of QC is the assumed value of sigma. Clusters with poor separation are discarded by keeping only the top xx% of best SSQs (in QC no solution is discarded because QC does not depend on random initialisation).

For each of the clusters retained at that value of K, the value of Cv is calculated from the contingency table of cluster allocation compared with each pair of clusters in turn. The median of this distribution of values of Cv is used as the characteristic measure of the stability of this cluster when measured against the other clusters.

The pairs of values (SSQ, median(Cv)) for all cluster solutions with high SSQ are then plotted, forming the SeCo map. This scatter plot gives useful information regarding values of K which represent the hierarchical data structure and, for each K, it identifies the most stable clustering solution.

The unsupervised performance assessment can be done following the next steps:

a) The first step is to run the QC over σ values between [0, 2] in regular intervals.
b) The second step is to measure the number of clusters per σ value; this information shows how the potential $V(\mathbf{x})$ evolves according to σ values, and reveals where the data structure is

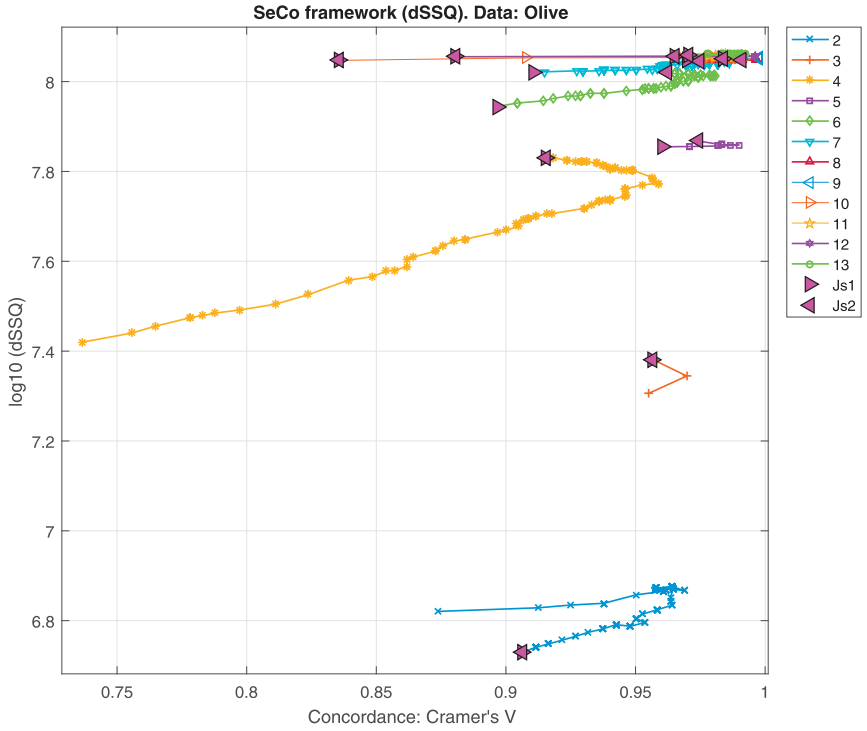


Fig. 7. SeCo representation for the olive oil data set. The plots contains the Separation and the Concordance for each number of clusters, marking the solution with the best Jaccard score for each cluster number, using two sets of external labels: Js1 stands for the solution corresponding with three regions and Js2 for the case of nine areas. For example, for cluster numbers $K = 3$ and 4, a better ΔSSQ score is achieved with the smallest value of σ for that K, which is highest on the y-axis. Although it is not obvious from the value of ΔSSQ, the best Jaccard score for 3 labels, Js1, occurs for $K = 4$. And the best for 9 labels, Js2, for $K = 8$.

more stable: the wider σ range the more stable data structure. Fig. 6 shows the solutions for the olive oil data set.

c) The third step is to obtain the SeCo framework. For every QC solution grouped by number of clusters, the ΔSSQ and the internal Concordance are calculated. The SeCo framework can be observed in Fig. 7 for the olive oil data set. Unfortunately, the graph needs to zoom in to appreciate each K in detail, and this justifies the plot of the next step.

d) In order to adapt the SeCo framework to QC, σ has been added as an additional variable in the SeCo framework. Plotting ΔSSQ against σ, and Concordance against σ, it is possible to observe all the relevant information in a straightforward way. Fig. 8 shows that representation for the olive oil data set.

Section 6 will show a deeper analysis of the procedure to select the most useful K and the corresponding solution. The process of finding a sufficiently good solution for unknown data consists in two parts; firstly, a selection of an appropriate K, and secondly, a solution within all K-groups' solutions. The criterion to select K has been based on choosing the lowest K (for simplicity) that improves considerably the Separation and has a good Concordance (not necessarily the best).

The external labels with the Jaccard scores can help to verify the conclusions obtained in this framework. For each K, three main solutions can be extracted, the solution with highest Separation, the solution with highest Concordance and the solution with highest Jaccard score (knowing the true labels). Comparing between them it is possible make an inference about which is the most relevant criterion. For instance, in Fig. 9 they can be compared for the olive oil data set, where one may see that the solution with the best Separation (ΔSSQ) is frequently almost as good as the one with the best Jaccard score.

6. Discussion of the experimental results

This section is focused on the SeCo vs. σ plots for the different data sets described in Section 3. The rest of the graphs presented in the previous section have been omitted to limit the length of the paper, and also because SeCo vs. σ is the most relevant plot in order to decide a useful K. To support the conclusions, the Jaccard score plots of the true labels are presented, as well.

6.1. Synthetic data sets

6.1.1. Artificial data set #1 (4 clusters)

This dataset is designed to produce a Concordance conflict when $K \neq 4$ because there are three groups of equidistant clusters, one group contains two close clusters and the other two have a single cluster. The Concordance conflict is due to different label assignments when they are equally probable. At least, this is the expected behaviour for K-Means.

Fig. 10 shows the SeCo vs σ. Here the expected conflict in Concordance is not as significant as it would be in K-Means. QC depicts the $K = 4$ as the widest σ range and it has constant high Concordance compared with other K values, what reveals the importance of the σ range as a cluster stability estimation.

In QC, the Concordance is not as relevant as it is in other algorithms because QC does not depend on random initialisations; every solution at σ_i is a slight variant of the solution at σ_{i-1} when σ values are sufficiently similar. The exception happens in the point

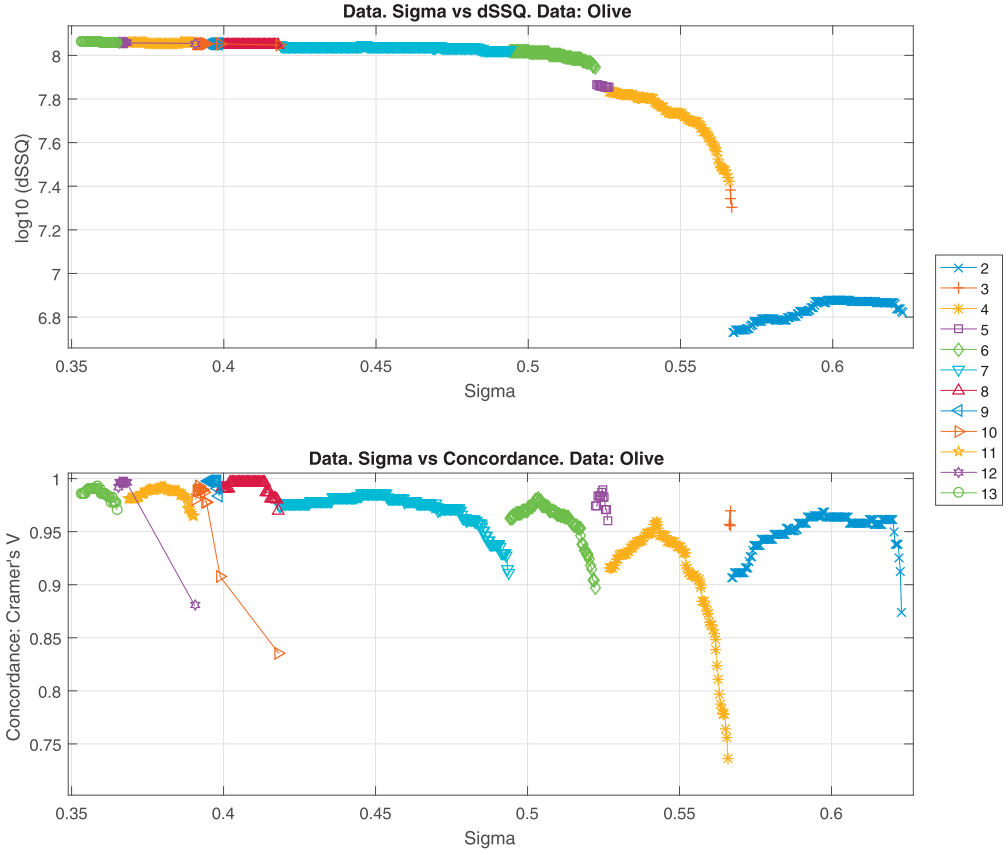


Fig. 8. SeCo vs. σ for the olive oil data set. Top graph shows ΔSSQ vs. σ. Bottom graph internal Concordance vs. σ. Additionally it is possible to appreciate the σ range as an estimation of cluster stability in the quantum potential $V(\mathbf{x})$. This figure illustrates three main points: First, there is a value of σ where the Separation stabilises. In this case it is $K = 6$. Second, within the range of K with high Separation, i.e. 6 and above, there is an increase in internal Concordance for $K = 8$. Thirdly, the value of K finally selected to be optimal for this data set, also has a wide range of values of σ. This confirms $K = 8$ in this case.

when the cluster number changes, then again the solutions evolve gradually till the next K.

Fig. 11 shows that any $K \geq 4$ is a suitable K; the solution with highest ΔSSQ has the same performance as the one with highest Concordance, given any K.

6.1.2. Artificial data set #2 (10 clusters)

In this data set, SeCo vs σ plot in Fig. 12 shows a curve plateau in the representation of ΔSSQ, thus suggesting that the solutions for $K \geq 4$ are quite separated. Attending the group Concordance, the best are $K \in [5, 8]$, all have a reasonably wide σ range compared with $K > 8$. Thus, the chosen solution should be one of the $K \in [5, 8]$, depending on the desirable number of clusters.

Comparing these results with the supervised Jaccard score plot in Fig. 13, it is observable the increasing performance with K even for values greater than the actual number of clusters, although the Jaccard score performance for $K \in [5, 8]$ is close to the plateau curve.

In any case, it is remarkable the poor performance in general of the QC for this dataset, with scores lower than 0.35. Other interesting aspect is the performance differences for the best Concordance solution and the best Separation solution with $K > 8$, being better for the latter.

6.2. Real data sets

6.2.1. Olive oil

Fig. 8 shows a considerable improvement in terms of ΔSSQ when the cluster number passes from $K = 2$ to 3 and so on, but ΔSSQ seems to stabilise in $K = 6$. This is a common pattern observed in all the tested datasets, there is a certain K where the ΔSSQ reaches the curve plateau. In this point QC already has found the main clusters, but beyond this point, more K only splits some clusters in additional subdivisions without really improving ΔSSQ. This is the main hint to indicate a suitable K beyond this point.

Next hints to pay attention are the internal Concordance and the σ range per K. From solutions with $K \geq 4$, the K with the highest internal Concordance as a group would be a good candidate. Next priority should be to give priority to those values of K with a wider σ range, for instance avoiding $K = 3$, 5, 9, 10 or 12. With those priorities, the best candidate should be $K = 8$. Regarding which is the best solution within $K = 8$, the priority should be the solution with highest ΔSSQ. This statement is based on the observation of the Jaccard score plots shown in Fig 9, where the solution with highest ΔSSQ is always closer to the best Jaccard score solution than the one with highest Concordance.

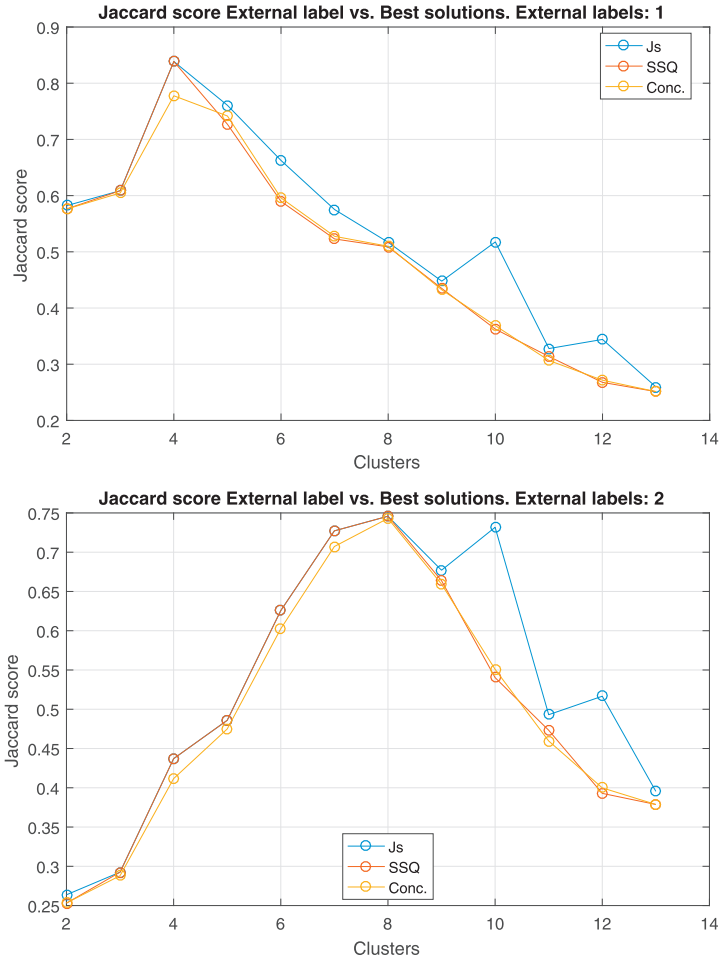


Fig. 9. Jaccard score for the two possible external labels in the olive oil data set: three regions (top plot) and nine sub-regions (bottom plot). The plots show three solutions, the one with best Jaccard score for that K, the one with best Separation and the one with best internal Concordance. These solutions may not be the same, i.e. they may have different values of σ for the same K. Refer to Fig. 7. This figure shows that, in general, a) the σ with best Separation (ΔSSQ) has a better Jaccard score than the σ with best Concordance, and b) the value of K selected by the proposed method, i.e. $K = 4$ and 8, have strong Jaccard scores.

6.2.2. Wine

The results of this dataset can be observed in Fig. 14; they are different from those obtained with the olive oil data set. The first difference is in the number of solutions, the main reason is due to the narrow σ range; for $K \in [2, 13]$, $\sigma \in [0.452, 0.493]$; this situation helps to explain how difficult is to find a suitable value of σ with the KNN approach.

Other aspects to remark are the unexpected valley in the ΔSSQ curve or the K fluctuation for adjacent σ values. These aspects point to a bad performance of QC in this dataset, and in fact, if one observes the Jaccard score (Fig. 15), a poor performance is observed. The wine data set is supposed to have easily separable clusters, but QC does not work well, probably because the hypersphere space transformation overlaps two labels when in the raw data it does not occur. In addition, the ratio observations/features is quite low, 174/13.

In any case, a suitable K based on SeCo vs σ plot would be $K = 8$ because it is the first K with high ΔSSQ and with good Concordance, despite its low σ range.

6.2.3. Iris

For the results corresponding to Iris data set, shown in Fig. 16, and following the ΔSSQ priority, the chosen K would be $K = 4$, which has a good Concordance and a wide σ range. However, Fig. 17 shows that the solutions for $K = 2$ or 3 have a higher Concordance and a higher Jaccard score than the chosen ones with $K \geq 4$, although this information would be unknown in an unsupervised scenario.

6.3. Summary of results

Table 1 summarises the main results obtained in the tested datasets. The ΔSSQ row indicates the chosen K according to the separation measure. The *Cv* row depicts those K which have the highest Cramér's V, revealing a possible underlying hierarchical structure. The *Js* row shows which K has the QC solution with the highest Jaccard score compared with the true labels, and finally, the last row shows the number of clusters of the true labels.

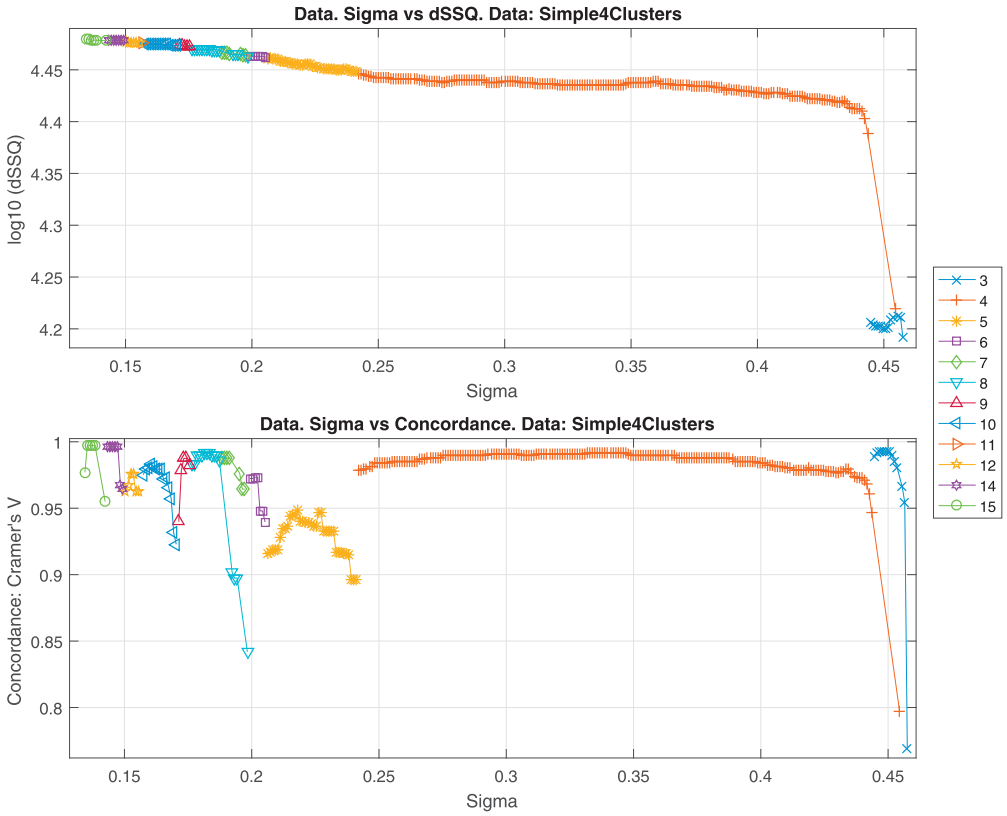


Fig. 10. SeCo vs. σ for the artificial data set #1 (4 clusters). Top graph shows ΔSSQ vs σ. Bottom graph internal Concordance vs. σ. Additionally it is possible to appreciate the σ range as an estimation of cluster stability in the quantum potential $V(\mathbf{x})$.

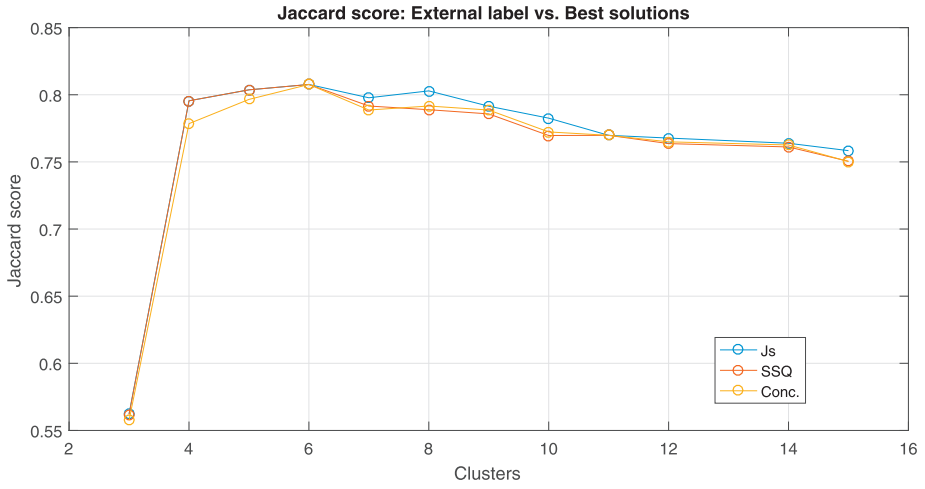


Fig. 11. Jaccard score for four external labels on artificial data set #1 (4 clusters). There are three solutions, the one with best Jaccard score for that K, the one with best Separation and the one with best internal Concordance.

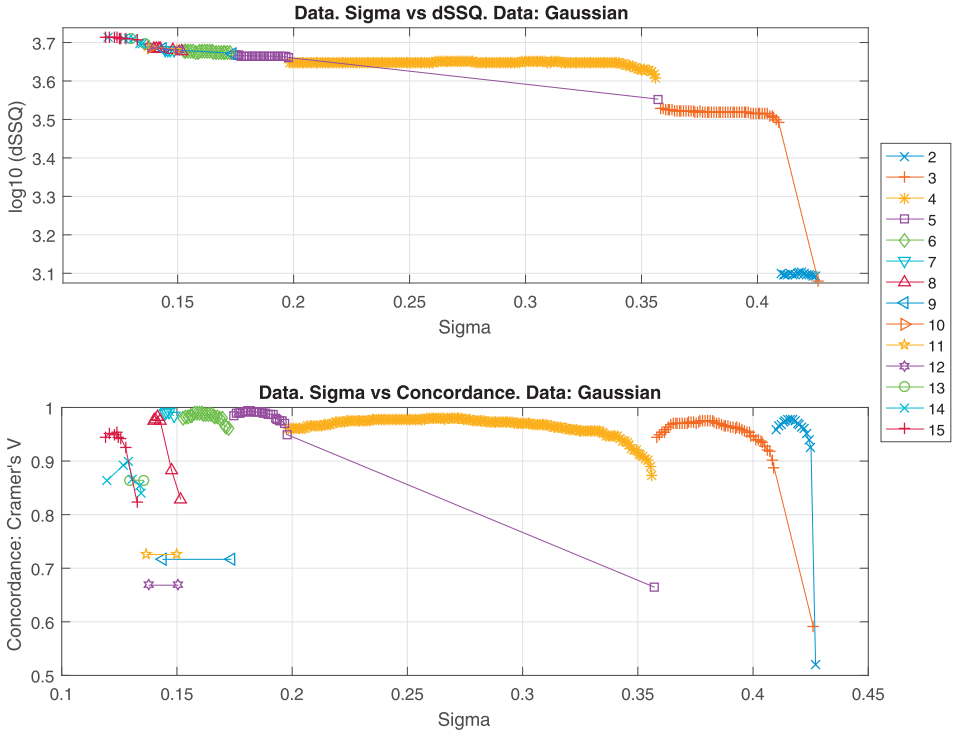


Fig. 12. SeCo vs. σ for the artificial data set #2 (10 clusters). Top graph shows ΔSSQ vs. σ. Bottom graph internal Concordance vs. σ. Additionally it is possible to appreciate the σ range as an estimation of cluster stability in the quantum potential $V(\mathbf{x})$.

Table 1
Cluster number K identified by different criteria with corresponding Jaccard scores against true labels shown within [ ].

Maximal criterion	Artificial dataset 1	Artificial dataset 2	Olive oil data	Wine data	Iris data
ΔSSQ	6 [0.81]	6 [0.34]	8 [0.75]	5 [0.4]	5 [0.49]
Cv	6 [0.81]	5 [0.32]	5 [0.70]	2, 3, 6	2 [0.58]
		6 [0.34]	8 [0.75]	All [0.34]	6 [0.44]
Js	≥ 4 [0.79–0.81]	10 [0.35]	4 [0.85]	4, 5, 7	2, 3
			vs 3 lab.	All [0.43]	Both [0.58]
			8 [0.75]		
			vs 9 lab.		
Number of true clusters	4	10	3, 9	3	4

6.4. Methodology to find the value of K

This section describes a schematic procedure for finding the most suitable K and its QC solution for unknown data.

The Algorithm 1 describes the methodology to find the SeCo parameters. Once the parameters have been obtained, the value of K with consistently best separation should be selected; then, the value of σ for the best separation is the QC of choice for that value of K. High values of the concordance measure for different number of clusters indicate the presence of a hierarchical structure.

Algorithm 1 Get N QC solutions and SeCo parameters.

$\sigma_N \leftarrow$ N-vector evenly distributed $\in]0, 2[$
for $i = 1 : N$ **do**
  $Solution_i \leftarrow QC(\sigma_i)$ ▷ Solution: QC outcome (vector of labels)
  $\Delta SSQ_i \leftarrow SSQ_{1cluster} - SSQ_{Solution_i}$
  $CN_i \leftarrow$ Cluster Number$(Solution_i)$
end for
Filter Solutions with $CN_i \in [CN_{min}, CN_{max}]$ ▷ To avoid excessive CN
for $CN_j = CN_{min} : CN_{max}$ **do**
  $Sol_j = Solutions$ with $CN = CN_j$
  for $i = 1 : max(Sol_j)$ **do**
    $CV_i = median(Cramer'sV(Sol_{ji}, Sol_{CN_j}))$ ▷ Internal Concordance
  end for
end for

6.5. Scalability

The problem of scalability falls on two main factors:

- Computational cost depends on the sample size (at least, linearly).
- It is also linear in the number of sigma values that need to be sampled to obtain a significant number of solutions in each cluster number, as required to properly estimate the median of Cv. In some data sets particular numbers of clusters only occur for a very narrow range of values of sigma, which makes this problem more complicated.

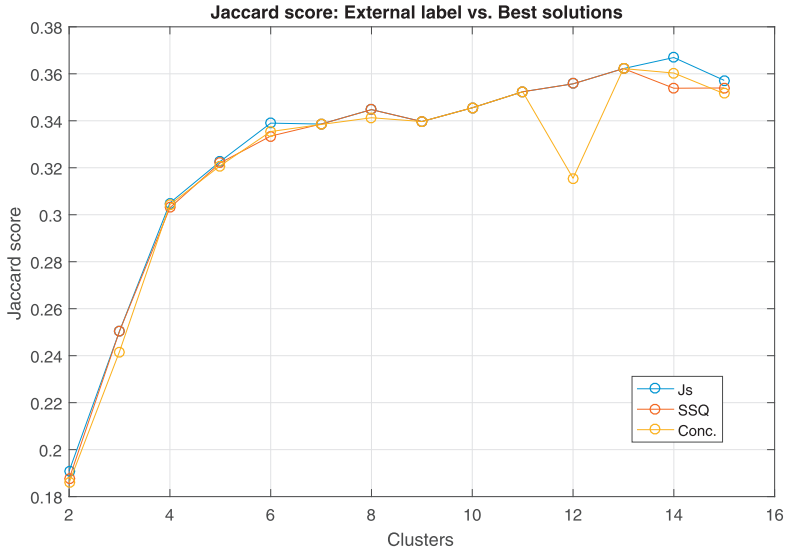


Fig. 13. Jaccard score for 10 external labels in the artificial data set #2 (10 clusters). There are three solutions, the one with best Jaccard score for that K, the one with best Separation and the one with best internal Concordance.

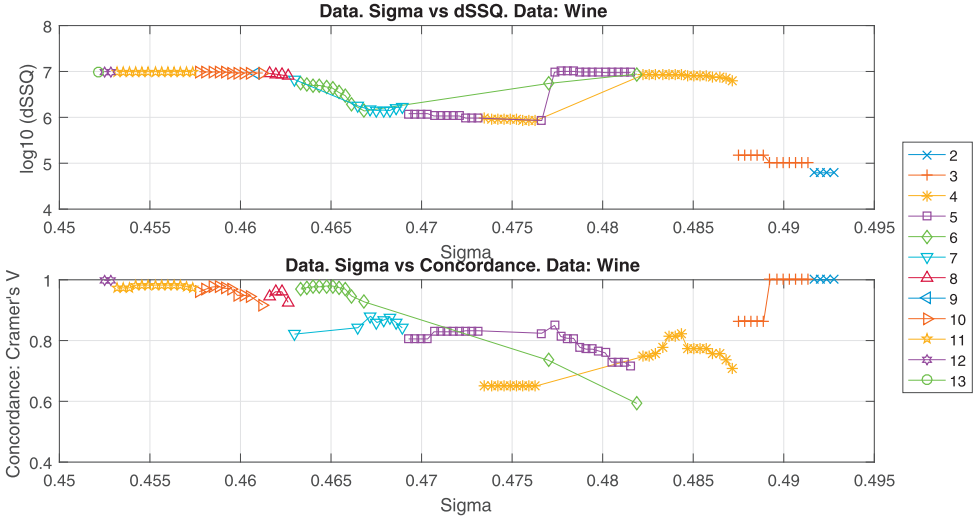


Fig. 14. SeCo vs. σ for the wine data set. Top graph shows ΔSSQ vs. σ; bottom graph internal Concordance vs. σ. Additionally it is possible to appreciate the σ range as an estimation of cluster stability in the quantum potential $V(\mathbf{x})$.

However, if data sets are very large then it may be possible to take random samples. In addition, it is not strictly necessary to sample every number of clusters.

Ultimately there is a clear need in QC for better ways to estimate the variance around data points and also for a more principled way to find the most likely number of cluster to best explain the data. This is indicated under further work.

In addition to this, in our view the main problem of QC is not related to large (if we understand by large to have many samples) datasets because there are straightforward ways to computational cost e.g. parallelisation. The main problem is more related with high dimensions and our current research partly deals with working on solutions for that. This applies data sets such as the cardiotocography sample from UCI.

Looking with a little more detail, QC depends on the length of the observations that generate the potential, n_{gen}, the number of points where the potential is computed, n_{alloc}, the dimension of the data, *dim*, and the number of steps applied in the stochastic gradient descent (SGD), $steps_{SGD}$. Therefore, an estimation of its time complexity is $O(n_{gen}{}^{*}n_{alloc}{}^{*}D^{*}steps_{SGD})$.

The SeCo framework increases the time complexity in a factor that depends on the number of different σ to be sampled, being a time complexity of $O(n_{gen} * n_{alloc} * D * steps_{SGD} * \#\sigma)$; in particular, 1,000 sigma samples have been used in the experiments presented in this paper. Nevertheless, this work has not been focused on time complexity or scalability, which are undoubtedly relevant topics for future research.

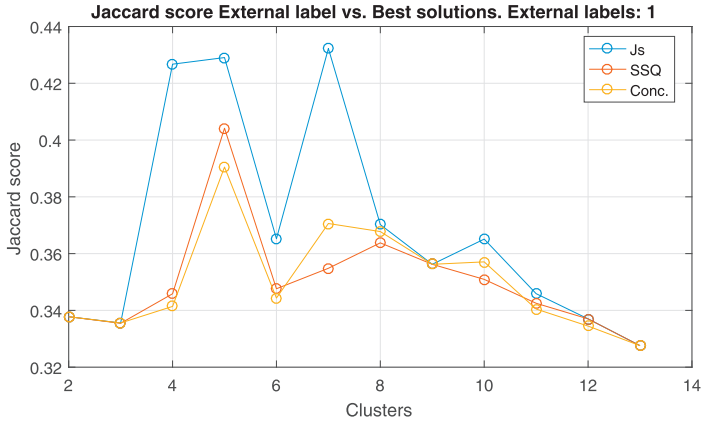


Fig. 15. Jaccard score for three external labels in the wine data set. There are three solutions, the one with best Jaccard score for that K, the one with best Separation and the one with best internal Concordance.

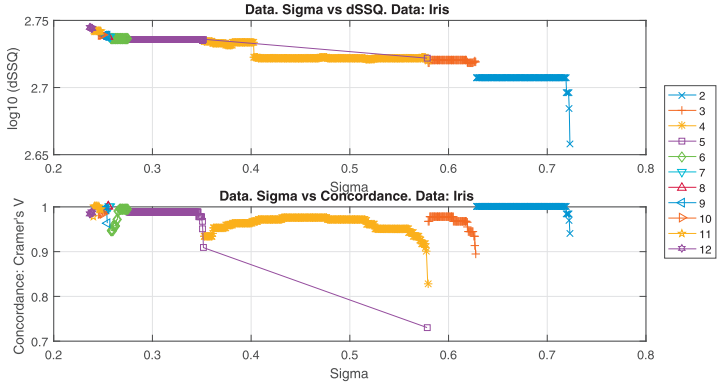


Fig. 16. SeCo vs. σ for the Iris data set. Top graph shows ΔSSQvs.σ. Bottom graph internal Concordance vs. σ. Additionally it is possible to appreciate the σ range as an estimation of cluster stability in the quantum potential $V(\mathbf{x})$.

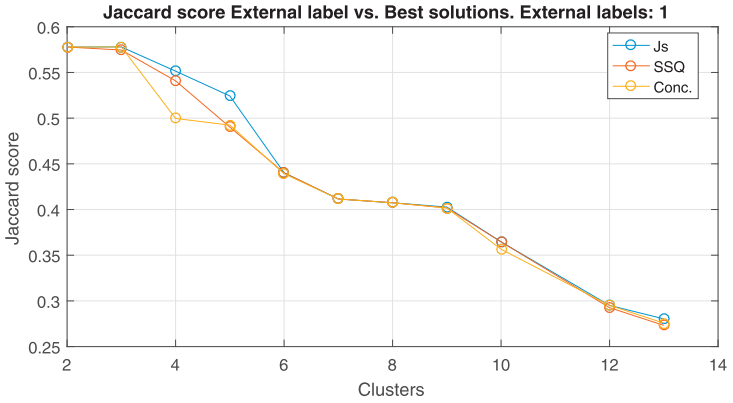


Fig. 17. Jaccard score for three external labels in the Iris data set. There are three solutions, the one with best Jaccard score for that K, the one with best Separation and the one with best internal Concordance.

7. Conclusions

This paper has proposed two figures of merit to characterise the quality of solutions obtained by QC, from which cluster numbers can be identified which maximise the fit against true cluster labels. Maximisation of cluster separation identifies clusters with consistently high Jaccard score against true labels, while high values of cluster consistency provide insights about hierarchical cluster structures. The proposed framework provides useful guidance to set an optimal value for the length scale parameter σ for each data set.

Two approaches have been proposed to find the best σ, and in turn, the correct number of clusters. The first one is based on the variance of K-Nearest Neighbours, and the second one on sampling QC solutions to apply the SeCo framework. The results yielded by the former approach do not suggest its use as a rule of thumb, due to three main reasons:

- It offers only one solution.
- The σ confidence intervals are too wide for the QC variability.
- The estimated σ strongly depends on the data structure, being very difficult to establish a general procedure.

The SeCo framework approach is based on measures of Separation (ΔSSQ), internal Concordance and σ range per cluster. The σ range parameter subtracts importance from the Concordance. Although the Concordance is very important in other algorithms like K-Means, it is less critical in QC due to its unique solution per σ value. The SeCo framework approach involves a higher computational cost than the KNN approach because of the need to run the algorithm multiple times. However, the results offer a consistent method to make a performance assessment in an unsupervised way. The SeCo plots have been adapted to the QC, adding an extra parameter: σ. The advantage of the SeCo vs σ plots is that they depict the data structure and the most suitable QC solutions in a straightforward way.

A procedure to select the appropriate K and the most suitable solution based on the empirical results has been proposed in Section 6.4.

Future work is needed to introduce better local tuning of the local variances across the data points, together with a principled approach to allocate points to clusters and for detection of outliers. Our current work is focused on how to tackle high dimensional data with QC where it loses performance.

A rigorous analysis of time complexity and scalability for complex data sets will also be considered in our future research.

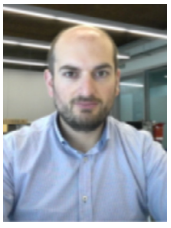
Acknowledgements

The authors thank Prof. David Horn, from the School of Physics and Astronomy (Tel Aviv University, Israel) for his valuable feedback.

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under project with reference number TIN2014-52033-R and by the Spanish Ministry of Education, Culture and Sport under project with reference number PR2015-00217.

References

[1] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth ed., Academic Press, 2008.
[2] P.J.G. Lisboa, T.A. Etchells, I.H. Jarman, S.J. Chambers, Finding reproducible cluster partitions for the k-means algorithm, BMC Bioinform. 14 (Suppl. 1) (2013) S8.
[3] S.J. Chambers, I.H. Jarman, T.A. Etchells, P.J.G. Lisboa, Inference of number of prototypes with a framework approach to k-means clustering, Int. J. Biomed. Eng. Technol. 13 (4) (2013) 323–340.
[4] S.J. Chambers, A Framework Approach to Initialisation Dependent Clustering, Liverpool John Moores University, UK, 2015 Ph.D. thesis.
[5] D. Horn, I. Axel, Novel clustering algorithm for microarray expression data in a truncated svd space, Bioinformatics 19 (9) (2003) 1110–1115.
[6] D. Horn, A. Gottlieb, Algorithm for data clustering in pattern recognition problems based on quantum mechanics, Phys. Rev. Lett. 88 (1–018702) (2002) 1–4.
[7] D. Horn, A. Gottlieb, The method of quantum clustering, in: Proceedings of Neural Information Processing Systems, NIPS 2001, 2001, pp. 769–776.
[8] R.V. Casaña-Eslava, J.D. Martín-Guerrero, I.H. Jarman, P.J.G. Lisboa, Performance assessment of quantum clustering in non-spherical distributions, in: Proceedings of the 24th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2016, pp. 339–344.
[9] J.D. Martín-Guerrero, A. Vellido, P.J.G. Lisboa, Physics and machine learning: emerging paradigms, in: Proceedings of the 24th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2016, pp. 319–326.
[10] N. Nasios, A.G. Bors, Finding the number of clusters for nonparametric segmentation, in: Proceedings of the 11th International Conference on Computer Analysis of Images and Patterns, CAIP„ 2005, pp. 213–221.
[11] N. Nasios, A.G. Bors, Kernel-based classification using quantum mechanics, Pattern Recognit. 40 (2006) 875–889.
[12] Y. Li, Y. Wang, Y. Wang, L. Jiao, Y. Liu, Quantum clustering using kernel entropy component analysis, Neurocomputing 202 (2016) 36–48.
[13] R. Shang, Z. Zhang, L. Jiao, W. Wang, S. Yang, Global discriminative-based nonnegative spectral clustering, Pattern Recognit. 55 (2016) 172–182.
[14] S.K. Tasoulis, D.K. Tasoulis, V.P. Plagianakos, Random direction divisive clustering, Pattern Recognit. Lett. 34 (2) (2013) 131–139.
[15] S. Tasoulis, L. Cheng, N. Välimäki, N.J. Croucher, S.R. Harris, W.P. Hanage, T. Roos, J. Corander, Random projection based clustering for population genomics, in: Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), IEEE, 2014, pp. 675–682.
[16] R. Varshavsky, M. Linial, D. Horn, COMPACT: a comparative package for clustering assessment, in: Proceedings of ISPA Workshops 2005, 2005, pp. 159–167. LNCS 3759
[17] Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml.
[18] D. Cook, D.F. Swayne, Interactive and Dynamic Graphics for Data Analysis, Springer Verlag, Berlin, Germany, 2007.
[19] M. Forina, C. Armanino, S. Lanteri, E. Tiscornia, Food Research and Data Analysis, Applied Science Publishers, pp. 189–214.



Raul V. Casaña-Eslava obtained Physics Degree in 2009 from University of Valencia (Spain). He spent five years working in industry as technologist and process engineer. Later he returned to the University of Valencia, where in 2015 he obtained a M.Sc. Degree in Electronics. During 2015/2016 he has been working as a researcher at Intelligent Data Analysis Laboratory (IDAL) in the University of Valencia (Spain). Nowadays he is a Ph.D. student in the Department of Applied Mathematics in Liverpool John Moores University (UK). His recent research is focused on Probabilistic Graphical Models and Quantum Clustering.

Dr. Ian Jarman is a Lecturer in the Department of Applied Mathematics. He is an expert in the analysis of large, high-dimensional data, using both standard statistical techniques and cutting edge non-linear algorithms for knowledge discovery and visualisation, and has extensive experience in multi-disciplinary collaborations in: Public Health, Medicine, Retail and Sports Science. He studied Mathematics, Statistics and Computing (1st Class, Honours) at Liverpool John Moores University, and took a PhD in Integrated frameworks for risk profiling of Breast Cancer patients in 2006.

Paulo Lisboa is Head of Department of Applied Mathematics and Head of the Research Centre for Data Science at Liverpool John Moores University. He has over 250 refereed publications, with awards for citations. His research focus is computer-based decision support in healthcare, sports analytics and computational marketing. In particular, he is interested in rigorous methods to interpret complex models for validation by subject area experts. He is chair of the Medical Data Analysis Task Force and co-chair of the Big Data Task Force in the Data Mining Technical Committee of the IEEE Computational Intelligence Society and past chair of the Healthcare Technologies Professional Network in the Institution of Engineering & Technology. He also has editorial and peer review roles in a number of journals and research funding bodies.

José D. Martín-Guerrero received a B.S. degree in Physics (1997), a B.S. degree in Electronic Engineering (1999), a M.S. Degree in Electronic Engineering (2001) and a Ph.D. degree in Machine Learning (2004), all from the University of Valencia, Spain. He is currently an Associate Professor at the Department of Electronic Engineering, University of Valencia. He also leads the Intelligent Data Analysis Laboratory, at the University of Valencia. His research interests include Machine Learning and Computational Intelligence, with special emphasis in Reinforcement Learning and Quantum Machine Learning. He is a founder Member of the Medical Data Analysis Task Force (Data Mining Technical Committee, IEEE Computational Intelligence Society).

Structure finding stabilization and optimization with the PC algorithm

Raúl V. Casaña-Eslava[(1)], Ian H. Jarman[(1)], Sandra Ortega-Martorell[(1)], Paulo J. Lisboa[(1)], José D. Martín-Guerrero[(2)]

(1) Department of Applied Mathematics, Liverpool John Moores University (LJMU)
r.v.casanaeslava@ljmu.ac.uk, i.h.jarman@ljmu.ac.uk, s.ortegamartorell@ljmu.ac.uk, p.j.lisboa@ljmu.ac.uk

(2) Departament d'Enginyeria Electrònica - ETSE, Universitat de València (UV)
jose.d.martin@uv.es

Keywords: Structure Finding, PC Algorithm, Parameter Selection, Directed Acyclic Graphs, Graph Node Ordering.

Abstract. In the field of structure finding, the PC algorithm is a well-known constraint-based algorithm used to build a Directed Acyclic Graph (DAG) from Conditional Independence maps where a major challenge is to minimize errors in the graph structure. This work presents empirical evidence for best practice: to reduce false positive errors via the False Discovery Rate (FDR), and to identify appropriate parameter settings to improve the False Negative Reduction (FNR). In addition, several node ordering policies are investigated that transform the skeleton graph into a DAG (edges orienting rules), the results show that ordering nodes by strength of mutual information recovers a representative DAG in reasonable time, although a more accurate graph can be recovered using a random order of samples at the expense of increasing the computation time.

1 Scientific Background

Structure finding methods lie within the field of Probabilistic Graphical Models and are deeply studied [Koller and Friedman, 2009, Daly et al., 2011], especially from a theoretical perspective, as they offer an efficient graphical approach to apply statistical estimates in a complex system. They serve as a framework for Bayesian and Markov networks [Bacciu et al., 2015], and have two components: a structure in the form of a graph, and a set of parameters that can be used to make statistical estimations.

Recently, a new structure finding algorithm [Bacciu et al., 2013] was proposed, which can obtain a faithful Bayesian Network (BN) without the need for specific approximations and with a reasonable computation time. Using these methods, complex associative maps can be obtained through Conditional Independence Maps (CI-Maps). Many constraint-based structure learning algorithms, including this work, are based on the PC-algorithm [Spirtes et al., 2000], where starting with a fully connected graph, edges are removed between nodes (variables) based on pairwise independence tests, increasing the number of conditioned variables as the algorithm progresses. The algorithm stops when it finally converges to a stable structure forming a CI-Map. This work extends the methods used in [Bacciu et al., 2013], where the criterion for independence is based on conditional mutual information instead of likelihood-ratio tests (G-test). In addition, the paper proposes several policies to create a data-driven structure:

- FDR [Benjamini and Hochberg, 1995] controls the False Positives decreasing the significance level in conditional independence tests when they are applied multiple times on the same nodes. This translates to reduced number of edges.

- FNR [Fast et al., 2008] tries to avoid independence tests if it is not powerful enough. The criteria is based on a threshold of Degrees of Freedom (DoF) that depends on the desired power for the test, the sample size and the effect size.

- The Weakest First (TWF) affects the order in which the graph is being pruned. The outcome of the PC algorithm is influenced by the order in which the conditional independence tests are executed, TWF sorts the nodes by mutual information (edge strength), testing first the weakest nodes which reduces the problem of incorrect pruning or incorrect edge dependence discovery.

Once the structure is found, the next step is to build a DAG following the rules defined in [Meek, 1995]. These rules do not necessarily lead to a unique DAG, where the most general solution is a Partial Directed Acyclic Graph (PDAG).

It is well known that the PC algorithm is sensitive to the order in which the nodes are tested [Kalisch and Bühlmann, 2008, Kalisch and Bühlmann, 2007]. This problem is addressed by using a similar solution to TWF, but ordering the nodes by descending mutual information. This policy is called The Strongest First (TSF).

Starting from the algorithm developed in [Bacciu et al., 2013], the aim of this work is to present empirical evidence for best practice for optimizing three parameters: the policies of *False Discovery Rate* and *False Negative Reduction*, and the *effect of node ordering* when the edges are being oriented. The work concludes with a final procedure for structure finding to address new data based on the parameters setup of the PC algorithm.

This work presents three novel contributions: 1) analysis of structure finding based on PC algorithm and its dependence on design parameters, e.g., data ordering; 2) empirical parameter optimization of conditional independence maps removing dependence on design parameters; 3) optimisation of FDR and FNR policies to avoid proliferation of False Positives (FPs).

2 Materials

Two datasets have been used for the benchmarking experiments, where the true DAGs are known: The ALARM and Insurance datasets, both available in Bayesian Network Repository[1]. ALARM stands for "A Logical Alarm Reduction Mechanism", and it is a network designed to provide alarms during patient monitoring in anaesthesia. This data has 37 variables (nodes), 46 edges degree (arcs), 509 parameters, 3.51 average Markov blanket size, 2.49 average degree and 4 maximum in-degree. The Insurance dataset is a Bayesian network for evaluating car insurance risks. It has 27 variables (nodes), 52 edges degree (arcs), 984 parameters, 5.19 average Markov blanket size, 3.85 average degree and 3 maximum in-degree. Samples of different size have been generated, from 0.5k to 100k observations.

3 Methodology

The methodology consists of two parts, analysing the structure errors of the CI-Maps, with reference to several set-ups of the FDR and FNR policies. Followed by the analyses of the influence of the node order when a DAG is created from a CI-Map.

When multiple conditional independence tests are carried out, the control of FDR is necessary. The work [Bacciu et al., 2013] proposes three different variants of the FDR control policy: *Basic*, *interleaved* and *mini-FDR*. *Basic* FDR is applied a posteriori after the PC algorithm converges. *Interleaved* is applied after each step of the PC algorithm, where a step implies one more node in the conditional independence tests. *Mini-FDR* prunes edges a priori, in the initial PC-algorithm stage.

[1] http://www.cs.huji.ac.il/~galel/Repository/

A False Negative (FN) implies a missing edge (independence) between two nodes (variables) when there is a true dependency present between these pair of nodes. It can produce collateral FPs if the CI-map tries to recover this lost dependency creating additional edges (FP) through the neighbour nodes. The FNR policy, based on [Fast et al., 2008], avoids testing the hypothesis if the minimum power of the test is not at least $1 - \beta$. This condition establishes a threshold for the maximum degrees of freedom that depends on the sample size, the false negative proportion β, the test level α, and a desired effect size w. While β and α are usually pre-established standards, the effect size w has to be adjusted, and it is difficult because the optimal value is dependent on the data characteristics and the sample size.

The BN can be evaluated by ordering the structure edges into a DAG, where BIC score is used for assessing the BN. As the order in which the edges are being oriented affects the final DAG, two predefined node orders by mutual information are proposed, comparing them to a random node order. Our conjecture is that they will produce more feasible BNs in terms of BIC score because the orientation decisions are taken first on the important edges. For the sake of completeness, we will also assess the opposite order, TWF, and the best and worst solutions created by random node order.

4 Results

Table 1 shows the structure errors for a 500 sample size of the Insurance dataset, using each of the FDR control policies. The FNR setup with an effect size of $w = 0.25$, and power of $\beta = 0.05$. The TWF node ordering is enabled. The table shows an improvement of five fewer FPs when FNR is activated, but no significant changes with reference to the FDR. Therefore the default policy, *basic FDR*, will be the recommended policy.

Table 1: Averaged structure errors of 10 samples of Insurance data with 500 observations each.

SETUP	*FN*	*FP*	*TOTAL*
FDR basic	3.1 ± 1.2	25.3 ± 1.5	28.4 ± 2.5
FDR interleaved	3.0 ± 1.5	25.2 ± 1.5	28.2 ± 2.8
FDR mini	2.8 ± 0.9	25.0 ± 1.6	27.8 ± 2.4
FDR basic + FNR	3.3 ± 0.9	20.6 ± 1.6	23.9 ± 1.9
FDR interleaved + FNR	3.4 ± 1.3	20.4 ± 1.4	23.8 ± 2.4
FDR mini + FNR	3.4 ± 1.1	20.4 ± 1.4	23.8 ± 2.3

Figure 1 shows the averaged structure errors of 10 different samples of Insurance dataset with 500 observations. This is an example to illustrate the pattern of the structure errors with reference to the effect size w. This behaviour occurs in a similar way for different sample sizes (not depicted in this work) but the graph is displaced to the left. FN errors decrease as the effect size decreases until some point where all tests are avoided and the graph becomes fully connected, beyond this point FNR is disabled. Similarly, where the FN errors start to decrease, the FP errors start to increase until reaching the fully connected graph where the FP errors are at a maximum value. There is an effect size window, δw, where the trade-off between FP vs FN is acceptable, and the total error decreases. This narrow δw depends on the dataset and the sample size. For this example, the best effect size value is $w_{min} = 0.24$, and improves the average total error by 4.7, not big improvements for the risk assumed out of δw.

Figure 2 shows the optimal effect sizes, w, found through empirical tests with Insurance (red) and Alarm (black) networks. The procedure was similar to that shown in figure 1, scanning several effect size values and selecting the one that minimizes the total structure errors. Additionally, in the same plot one may observe in blue the effect size suggested by [Fast et al., 2008]. The variability of the optimal effect size with the

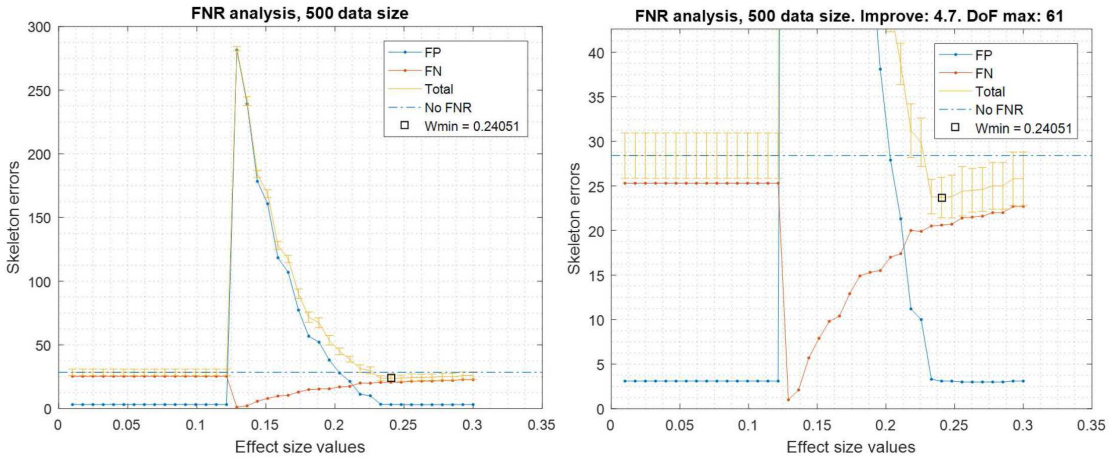


Figure 1: Structure errors of 10 averaged Insurance datasets with 500 observations, zoomed on the critical point.

data makes it very hard to set a rule of thumb to estimate effect sizes for unknown data.

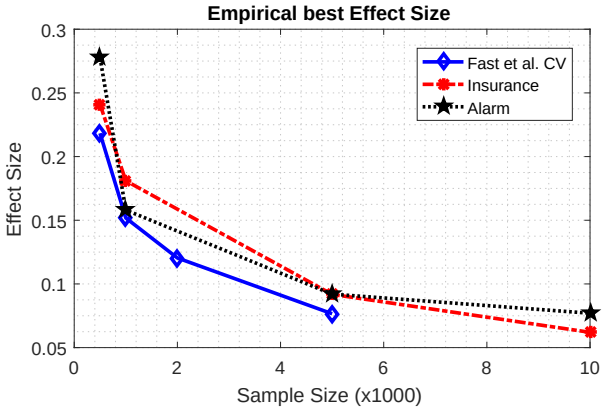


Figure 2: Optimal effect size parameter based on experimental results for Insurance (red) and Alarm data (black). The values are compared with the averaged results (blue) suggested in [Fast et al., 2008] applying a cross-validation on Insurance and Mildew data.

Figure 3 shows how a fixed effect size w affects the structure errors when the data size increases. When FNR is activated an increasing FP error is observed when the data size increases. If FNR is deactivated a significant FP reduction is observed. Theoretically, the PC-algorithm would converge to the true structure if the whole population data were available, however in a realistic case scenario the errors asymptotically tend to a non zero value, ≈ 7.5.

Figure 4 shows the density functions of the BIC score, using a non-parametric kernel-smoothing distribution for 100 samples of the Insurance dataset with 25k observations and 27 nodes. In each plot there are four distributions, the ones with TWF and TSF node order, and the best/worst solutions obtained by random ordering. There are three scenarios: only one random order, 25 and 100 random samples. It is shown that TSF is the best option when there is a limitation in the number of repetitions. This limitation may be due to a computational time constraint in big datasets, or when the graph has too many nodes and a large number of repetitions is needed to extract a representative sample of the node order permutations. If a high number of random order iterations is possible, the best option will be to obtain the BN with best BIC score.

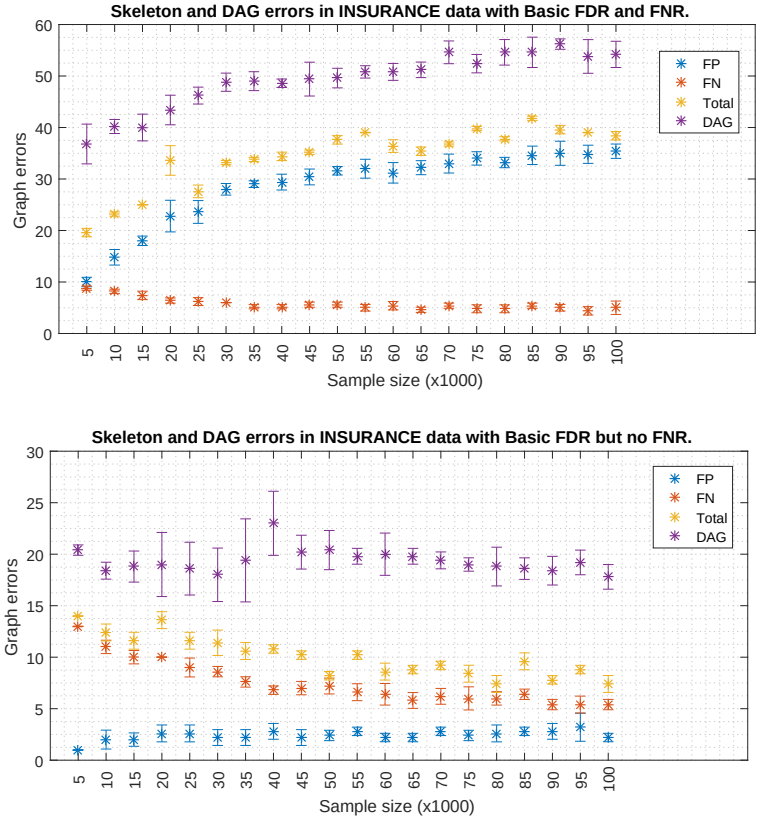


Figure 3: Errors in the skeleton and in the DAG when the sample size increases but the effect size of the FNR policy is fixed, FPs increase with the sample size for the Insurance data. FNs decrease in both cases, but if FNR is deactivated, FPs errors keep low and stable.

5 Conclusion

This work has been focused on finding the best setup for structure finding stabilization based on the PC algorithm. Any FDR policy can be used to control the FP errors, but the basic FDR has been selected due to its simpler implementation. FNR policy is focused on decreasing the FN errors, but it is not recommended when the optimal values of the effect size are unknown, which is the usual case for new data. A wrong effect size parameter in FNR policy can produce a proliferation of FP errors. Finally, to build a DAG from the structure, the node in which the edges are oriented affects the final DAG. We have proposed ordering the nodes by mutual information followed by orientating by the TSF order which provided the best BIC score compared with a random ordering distribution. However, if the DAG is generated multiple times by sampling random node orders, the best solution of multiple random node orders outperforms the TSF order.

References

[Bacciu et al., 2013] Bacciu, D., Etchells, T. A., Lisboa, P. J., and Whittaker, J. (2013). Efficient identification of independence networks using mutual information. *Computational Statistics*, 28(2):621–646.

[Bacciu et al., 2015] Bacciu, D., Lisboa, P. J., Sperduti, A., and Villmann, T. (2015). Probabilistic modeling in machine learning. In *Springer Handbook of Computational Intelligence*, pages 545–575. Springer.

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300.

[Daly et al., 2011] Daly, R., Shen, Q., and Aitken, S. (2011). Learning bayesian networks: approaches and issues. *The knowledge engineering review*, 26(2):99–157.

[Fast et al., 2008] Fast, A., Hay, M., and Jensen, D. (2008). Improving accuracy of constraint-based structure learning. Technical report, Technical report 08-48, University of Massachusetts Amherst, Computer Science Department.

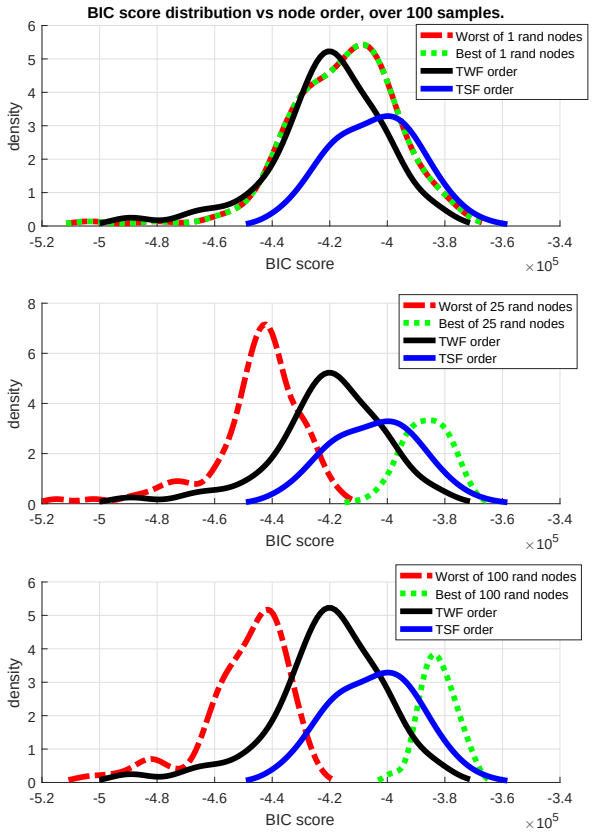


Figure 4: BIC score of node order distributions, over 100 samples of Insurance network with 25k observations. The TWF/TSF have a fixed order, then for 1, 25, 100 iterations of random node order are selected the best/worst solutions. Obviously, for one order, the best and the worst solutions are the same.

[Kalisch and Bühlmann, 2007] Kalisch, M. and Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.*, 8:613–636.

[Kalisch and Bühlmann, 2008] Kalisch, M. and Bühlmann, P. (2008). Robustification of the pc-algorithm for directed acyclic graphs. *Journal of Computational and Graphical Statistics*, 17(4):773–789.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

[Meek, 1995] Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc.

[Spirtes et al., 2000] Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.