DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

Facultad de Informática
Universidad Politécnica de Madrid

PhD THESIS

Semi-supervised subspace clustering and applications to neuroscience

Author

Luis Guerra
MS Computer Science
MS Artificial Intelligence

PhD supervisors

Concha Bielza
PhD Computer Science

Víctor Robles
PhD Computer Science

2012

Thesis Committee

President:

External Member:

Member:

Member:

Secretary:

La constancia es el
complemento ideal para el
resto de virtudes humanas

Acknowledgements

I intend to write a brief acknowledgement section but covering all the people who have been important to finish this thesis.

Foremost, I would like to thank my supervisors Concha Bielza and Víctor Robles, and also Pedro Larrañaga, for these years of intense work, full of meetings, reviews and, mainly, learning moments. Not only this work, but also myself, would not be the same without all these referred moments.

Also related to this work, and more specifically to the interdisciplinary aspect, I would like to thank Javier DeFelipe, Rafael Yuste and Ruth Benavides-Piccione, for their patience at teaching some of their huge knowledge about neuroscience to people like me, who knew almost nothing about this amazing field.

I also want to thank Nahid Shahmehri and Jose M. Peña not only for giving me the opportunity of visiting their group, but also for their hospitality, support with all the stuff that arose at that time, and also their advice, during my stay in Sweden at Linköping University with the Division for Database and Information Techniques.

Besides all these people, I want to thank all the managers and members of the Cajal Blue Brain Project (some of them were previously named) since this thesis would not have been possible without the economical and structural support of this project.

I want to begin the less formal part of this section thanking all my friends, colleagues, and teammates who have been with me throughout all these years sharing their lives. I do not want to be more specific because I certainly know that they all recognise who they are.

Last but of course not least, I wish to thank my family, specially my parents and my sister, for all the support and love, and also for being the best role models to follow in my life. To them I dedicate this work.

Abstract

Machine learning techniques are used for extracting valuable knowledge from data. Nowadays, these techniques are becoming even more important due to the evolution in data acquisition and storage, which is leading to data with different characteristics that must be exploited. Therefore, advances in data collection must be accompanied with advances in machine learning techniques to solve new challenges that might arise, on both academic and real applications.

There are several machine learning techniques depending on both data characteristics and purpose. Unsupervised classification or clustering is one of the most known techniques when data lack of supervision (unlabeled data) and the aim is to discover data groups (clusters) according to their similarity. On the other hand, supervised classification needs data with supervision (labeled data) and its aim is to make predictions about labels of new data. The presence of data labels is a very important characteristic that guides not only the learning task but also other related tasks such as validation.

When only some of the available data are labeled whereas the others remain unlabeled (partially labeled data), neither clustering nor supervised classification can be used. This scenario, which is becoming common nowadays because of labeling process ignorance or cost, is tackled with semi-supervised learning techniques. This thesis focuses on the branch of semi-supervised learning closest to clustering, i.e., to discover clusters using available labels as support to guide and improve the clustering process.

Another important data characteristic, different from the presence of data labels, is the relevance or not of data features. Data are characterized by features, but it is possible that not all of them are relevant, or equally relevant, for the learning process. A recent clustering tendency, related to data relevance and called subspace clustering, claims that different clusters might be described by different feature subsets. This differs from traditional solutions to data relevance problem, where a single feature subset (usually the complete set of original features) is found and used to perform the clustering process.

The proximity of this work to clustering leads to the first goal of this thesis. As commented above, clustering validation is a difficult task due to the absence of data labels. Although there are many indices that can be used to assess the quality of clustering solutions, these validations depend on clustering algorithms and data characteristics. Hence, in the first goal three known clustering algorithms are used to cluster data with outliers and noise, to critically study how some of the most known validation indices behave.

The main goal of this work is however to combine semi-supervised clustering with subspace clustering to obtain clustering solutions that can be correctly validated by using either known indices or expert opinions. Two different algorithms are proposed from different points of view to discover clusters characterized by different subspaces. For the first algorithm, available data labels are used for searching for subspaces firstly, before searching for clusters. This algorithm assigns each instance to only one cluster (hard clustering) and is based on mapping known labels to subspaces using supervised classification techniques. Subspaces are then used to find clusters using traditional clustering techniques. The second algorithm uses available

data labels to search for subspaces and clusters at the same time in an iterative process. This algorithm assigns each instance to each cluster based on a membership probability (soft clustering) and is based on integrating known labels and the search for subspaces into a model-based clustering approach. The different proposals are tested using different real and synthetic databases, and comparisons to other methods are also included when appropriate.

Finally, as an example of real and current application, different machine learning techniques, including one of the proposals of this work (the most sophisticated one) are applied to a task of one of the most challenging biological problems nowadays, the human brain modeling. Specifically, expert neuroscientists do not agree with a neuron classification for the brain cortex, which makes impossible not only any modeling attempt but also the day-to-day work without a common way to name neurons. Therefore, machine learning techniques may help to get an accepted solution to this problem, which can be an important milestone for future research in neuroscience.

Resumen

Las técnicas de aprendizaje automático se usan para extraer información valiosa de datos. Hoy en día, la importancia de estas técnicas está siendo incluso mayor, debido a que la evolución en la adquisición y almacenamiento de datos está llevando a datos con diferentes características que deben ser explotadas. Por lo tanto, los avances en la recolección de datos deben ir ligados a avances en las técnicas de aprendizaje automático para resolver nuevos retos que pueden aparecer, tanto en aplicaciones académicas como reales.

Existen varias técnicas de aprendizaje automático dependiendo de las características de los datos y del propósito. La clasificación no supervisada o *clustering* es una de las técnicas más conocidas cuando los datos carecen de supervisión (datos sin etiqueta), siendo el objetivo descubrir nuevos grupos (agrupaciones) dependiendo de la similitud de los datos. Por otra parte, la clasificación supervisada necesita datos con supervisión (datos etiquetados) y su objetivo es realizar predicciones sobre las etiquetas de nuevos datos. La presencia de las etiquetas es una característica muy importante que guía no solo el aprendizaje sino también otras tareas relacionadas como la validación.

Cuando solo algunos de los datos disponibles están etiquetados, mientras que el resto permanece sin etiqueta (datos parcialmente etiquetados), ni el *clustering* ni la clasificación supervisada se pueden utilizar. Este escenario, que está llegando a ser común hoy en día debido a la ignorancia o el coste del proceso de etiquetado, es abordado utilizando técnicas de aprendizaje semi-supervisadas. Esta tesis trata la rama del aprendizaje semi-supervisado más cercana al *clustering*, es decir, descubrir agrupaciones utilizando las etiquetas disponibles como apoyo para guiar y mejorar el proceso de *clustering*.

Otra característica importante de los datos, distinta de la presencia de etiquetas, es la relevancia o no de los atributos de los datos. Los datos se caracterizan por atributos, pero es posible que no todos ellos sean relevantes, o igualmente relevantes, para el proceso de aprendizaje. Una tendencia reciente en *clustering*, relacionada con la relevancia de los datos y llamada *clustering* en subespacios, afirma que agrupaciones diferentes pueden estar descritas por subconjuntos de atributos diferentes. Esto difiere de las soluciones tradicionales para el problema de la relevancia de los datos, en las que se busca un único subconjunto de atributos (normalmente el conjunto original de atributos) y se utiliza para realizar el proceso de *clustering*.

La cercanía de este trabajo con el *clustering* lleva al primer objetivo de la tesis. Como se ha comentado previamente, la validación en *clustering* es una tarea difícil debido a la ausencia de etiquetas. Aunque existen muchos índices que pueden usarse para evaluar la calidad de las soluciones de *clustering*, estas validaciones dependen de los algoritmos de *clustering* utilizados y de las características de los datos. Por lo tanto, en el primer objetivo tres conocidos algoritmos se usan para agrupar datos con valores atípicos y ruido para estudiar de forma crítica cómo se comportan algunos de los índices de validación más conocidos.

El objetivo principal de este trabajo sin embargo es combinar *clustering* semi-supervisado con *clustering* en subespacios para obtener soluciones de *clustering* que puedan ser validadas de forma correcta utilizando índices conocidos u opiniones expertas. Se proponen dos algo-

ritmos desde dos puntos de vista diferentes para descubrir agrupaciones caracterizadas por diferentes subespacios. Para el primer algoritmo, las etiquetas disponibles se usan para buscar en primer lugar los subespacios antes de buscar las agrupaciones. Este algoritmo asigna cada instancia a un único *cluster* (*hard clustering*) y se basa en mapear las etiquetas conocidas a subespacios utilizando técnicas de clasificación supervisada. El segundo algoritmo utiliza las etiquetas disponibles para buscar de forma simultánea los subespacios y las agrupaciones en un proceso iterativo. Este algoritmo asigna cada instancia a cada *cluster* con una probabilidad de pertenencia (*soft clustering*) y se basa en integrar las etiquetas conocidas y la búsqueda en subespacios dentro de *clustering* basado en modelos. Las propuestas son probadas utilizando diferentes bases de datos reales y sintéticas, incluyendo comparaciones con otros métodos cuando resulten apropiadas.

Finalmente, a modo de ejemplo de una aplicación real y actual, se aplican diferentes técnicas de aprendizaje automático, incluyendo una de las propuestas de este trabajo (la más sofisticada) a una tarea de uno de los problemas biológicos más desafiantes hoy en día, el modelado del cerebro humano. Específicamente, expertos neurocientíficos no se ponen de acuerdo en una clasificación de neuronas para la corteza cerebral, lo que imposibilita no sólo cualquier intento de modelado sino también el trabajo del día a día al no tener una forma estándar de llamar a las neuronas. Por lo tanto, las técnicas de aprendizaje automático pueden ayudar a conseguir una solución aceptada para este problema, lo cual puede ser un importante hito para investigaciones futuras en neurociencia.

Contents

Acronyms

3SMBC semi-supervised subspace model-based clustering

AIC Akaike's information criterion

ARI adjusted Rand index

B `basket`

BBP Blue Brain Project

BIC Bayesian information criterion

BSS between-cluster sum of squares

CBBP Cajal Blue Brain Project

CDCDD constraint based dimension correlation and distance divergence

CLIQUE clustering in quest

CLWC constrained locally weighted clustering

COSA clustering on subsets of attributes

CQI clustering quality index

CRISP-DM cross-industry standard process for data mining

CSIC Consejo Superior de Investigaciones Científicas

CTh candidates threshold

DB Davies-Bouldin

DBSCAN density-based spatial clustering of applications with noise

DiSH detecting subspace cluster hierarchies

DOC density-based optimal projective clustering

EA evolutionary algorithm

EM expectation-maximization

ENCLUS entropy-based clustering

EPFL École Polytechnique Fédérale de Lausanne

FINDIT fast and intelligent subspace clustering algorithm using dimension voting

FIRES filter refinement subspace clustering

FN false negative

FP false positive

FSE feature subset extraction

FSS feature subset selection

GA genetic algorithm

HARP hierarchical approach with automatic relevant dimension selection for projected clustering

HMRF hidden Markov random field

HT `horse-tail`

IC Instituto Cajal

KDD knowledge discovery in databases

KL Kullback-Leibler

KMF knowledge mapping framework

K-nn K-nearest neighbors

LAC locally adaptive clustering

LR logistic regression

MAFIA merging of adaptive finite intervals

MLP multilayer perceptron

MML minimum message length

MPCKM metric pairwise constrained K-means

MT `Martinotti`

NB naïve Bayes

ORCLUS oriented projected cluster generation

P3C projected clustering via cluster cores

PCA principal component analysis

PC principal component

PLS partial least squares

PreDeCon preference weighted density connected clustering

PROCLUS projected clustering

SCHISM support and Chernoff-Hoeffding bound-based interesting subspace miner

SCKMM semi-supervised clustering Kernel method based on metric learning

SISC semi-supervised impurity based subspace clustering

SSPC semi-supervised projected clustering

SUBCLU density-connected subspace clustering

SVM support vector machine

TAN tree augmented naïve Bayes

TN true negative

TP true positive

UPM Universidad Politécnica de Madrid

WSS within-cluster sum of squares

Part I

INTRODUCTION

Chapter 1

Introduction

With constant advances in both computer technology and data acquisition, the amount of data that can be not only stored but also processed is getting larger and larger. Due to the vast amount, and also the complexity of data, human beings cannot analyze the information directly, and machine learning techniques are used for extracting valuable knowledge from data. Therefore, advances in machine learning techniques must be linked to available data and their characteristics, adapting and creating algorithms depending on data requirements.

There are several machine learning tasks, but traditionally, two of the most known are supervised classification and unsupervised classification or clustering. These two tasks are different depending on both input data characteristics and purpose, and their understanding is necessary to locate the work presented in this thesis.

Regarding data characteristics, a data set is composed of single instances. Each instance is uniquely identified by a descriptive feature vector, where each feature is a random variable that represents a measurement about a characteristic or an event related to the instance. Besides, an instance may be also identified by a class variable, also called label or merely class, that represents the classification of an instance. This label guides the learning process when available, and is the essential difference between data in supervised classification and clustering. Thus, in supervised classification each single instance is characterized by a pair of features vector and class variable, and only by a feature vector, without any kind of supervision, in unsupervised classification.

Regarding the purpose, the goal of supervised classification is learning a model according to some instances, which are called training data. This model is then used to make label predictions, based only on the known ones, about new instances that might appear without labels. Therefore, supervised classification is a predictive task. On the other hand, the goal of clustering is to find hidden structures in data, grouping instances based on their feature vectors. Therefore, clustering is a descriptive task. There are some others, but supervised classification and clustering are pattern recognition tasks both aiming at finding classes (labels or groups) for instances.

Although supervised classification and clustering must be seen as different tasks and comparisons between them should be made wisely, the lack of labels in clustering, and therefore of

supervision, turns some related processes, like validation, into a more difficult process than in supervised classification. Therefore, data labels, when available, might provide an important knowledge that can be used, not only for validating, but also for helping during the learning process.

Semi-supervised learning

The importance of data labels is evident, since their presence is one of the main reasons to select the pattern recognition task. The main problem related to labels is they are still scarce in some domains due to different reasons. On some occasions this absence is due to the label obtaining cost, either in money or in personnel. The data gathering procedure may have a high cost depending on the domain, like in neuroscience or genomics. Nevertheless, this is not the most challenging situation, since on other occasions, the absence of data labels is directly due to the ignorance about them. Hence, pattern recognition techniques become even more important concerns in that case.

When the absence of data labels is complete, i.e., a data set with all unlabeled instances, clustering is used as pattern recognition task as commented above. However, between the situations of total ignorance and total knowledge about data labels, there is a possible intermediate situation where labels for some instances are known, but not for the remaining ones. This kind of data is called partially labeled data.

When dealing with a partially labeled data, and taking into account the previously presented tasks, there are three different situations:

- To use a supervised classification approach. This situation might be considered when the number of labeled instances is high enough. Therefore, a model is learned from the labeled instances and the remaining labels are induced from the model. When the number of labeled instances is not high enough, the model that can be obtained could not be accurate or even usable. Note that predicted data labels for unlabeled data will be based only on the known labels from labeled data.

- To use a clustering approach. If the number of labeled instances is very low or even when the user does not matter to discard them, clustering might be used as if a completely unlabeled data set were available. Therefore, available data labels are discarded and are not used during the learning process. Nevertheless, labels are not completely useless in this choice since they can be used to validate the obtained results, evaluating whether the found groups are consistent with the data labels.

- Rigorously based on the input data characteristics of each task, neither supervised classification nor clustering can be used as approaches to solve the learning task. This fact leads to an intermediate task called semi-supervised learning.

Semi-supervised learning can be used when available data have some partial supervision, like when dealing with partially labeled data. Depending on whether the unlabeled instances

can be classified into one of the known labels or if there is the possibility of discovering new previously unknown labels, the task can be called semi-supervised classification or **semi-supervised clustering**, respectively. A common characteristic of these two tasks is that all available instances are somehow used to perform the learning process. Thus, if learning is based on some supervised classification algorithm, unlabeled data are employed to enhance the model. On the other hand, if learning is based on some clustering algorithm, labeled data are used trying to improve the obtained results.

The summary of where semi-supervised classification and semi-supervised clustering are situated depending on input data type and purpose, and also with respect to supervised classification and clustering, can be seen in Figure 1.1.

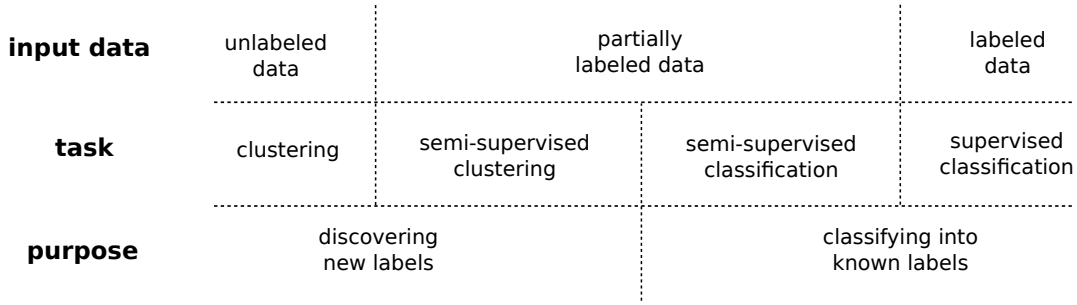


Figure 1.1: Pattern recognition tasks depending on data and task purpose.

Subspace search

Considering again the huge amount of available data, and independently of the task, the available number of features may become very high. This evidence turns the solution process more complicated and, even in some cases, unsolvable. For this reason, a dimensionality reduction if often carried out before or together with the learning task, with the aim of obtaining not only more accurate models, but also simpler. Simplest models are preferable in order to satisfy the well-known Occam's razor [32], which recommends the use of parsimonious models.

Dimensionality reduction can be achieved either by selecting a subset of features or by extracting new features from the available ones. The first approach is a process called feature subset selection (FSS), whereas the second approach is called feature subset extraction (FSE). Traditionally, both approaches aim at obtaining a single subset of features to perform the learning task. Therefore, there is a global dimensionality reduction. One of the main disadvantages of FSE is that new extracted features can be difficult to interpret when they must be explained. For this reason, FSS may be preferable although both approaches are commonly used

FSS is, similarly to validation, often easier when data labels are available because they can be used to guide the selection process. Maybe for this reason, FSS has been widely studied for supervised classification. However, FSS in clustering is still an open topic that

requires many improvements. This fact has not prevented that a new approach, related to FSS and called **subspace clustering**, has emerged for clustering tasks.

In clustering task, the found subset of features is used for finding hidden structures in data. Nevertheless, it is possible that using a single subset of features, some hidden structures in data remain hidden. Hence different subsets of features would be necessary to find all the structures in data. Therefore, there are several local dimensionality reductions. Subspace clustering is a tendency based on this assumption that has been successfully applied in clustering problems recently.

The summary of where the search for subspaces is situated depending on how the subsets of features are obtained and the number of them, and also with respect to other known dimensionality reduction processes, can be seen in Figure 1.2.

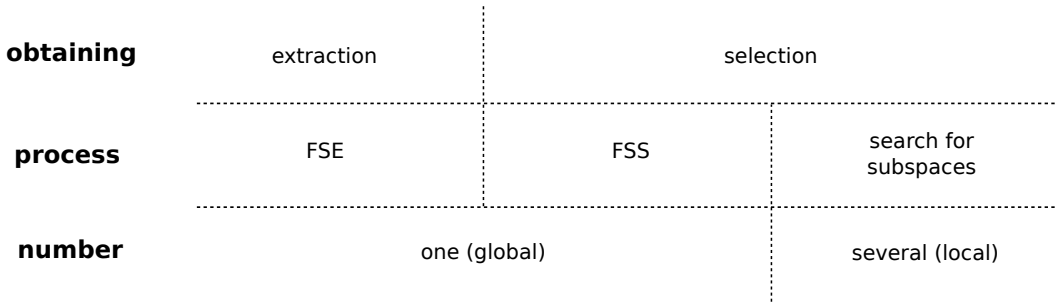


Figure 1.2: Dimensionality reduction processes depending on how the subsets of features are obtained and the number of them.

Applications

From this introduction, the close relation between data and machine learning is evident. There is not only (and at least) one machine learning task to solve each learning problem that can arise from data, but also machine learning tasks are constantly evolving either to solve new problems that can emerge or to obtain more accurate solutions for the known ones. Besides, machine learning is applied in very different fields. One of the best examples is "bioinformatics": machine learning is very often applied to solve biological problems, such as genomics or proteomics. Following this guideline, and also related to biology, neuroscience is one of the fields that is receiving increasing amounts of attention nowadays.

Chapter outline

This chapter continues with the presentation of the evidences that can be extracted from this introduction and the main motivations for this thesis. Then, the hypothesis and objectives are presented in Section 1.2. Finally the complete organization of this manuscript is detailed in Section 1.3.

1.1 Evidences and motivation

The research of this thesis is centered on the topics presented in the previous introduction, focusing on clustering validation and mainly on semi-supervised clustering problems together with subspace clustering. Besides, an important aspect of this research is the application of machine learning in general, and the obtained advances in particular, to neuroscience, as an example of challenging and real task. Before detailing the motivations for these topics, a series of evidences, extracted from the previous introduction, are presented:

- Clustering algorithms must be used when the aim is to find new and hidden groups in data, taking into account the difficulty of processes as validation due to the absence of labels.
- Data labels, when available, might help not only to validate results, but also to find them. When data labels are used together with unlabeled data in a clustering task, it is called semi-supervised clustering.
- Due to possible huge amounts of features available, FSS process is often necessary to find more accurate and simpler solutions.
- Related to the previous evidence, data groups can be hidden in different data dimensions in clustering. Therefore, subspace clustering, where different feature subsets are searched to characterize each group, can be used to find hidden data structures.
- There are many interdisciplinary tasks in which data mining might help to find solutions for real problems in fields like neuroscience.

The main motivation for this thesis is to extract valuable knowledge from partially labeled data. This kind of data arises in challenging problems, mainly when instances are not labeled due to ignorance about their labels and also when completely unknown groups, which do not appear in the labeled instances, may be discovered.

1.2 Hypothesis and objectives

Based on the evidences and motivation presented above, this research has a main and decomposable hypothesis:

- Semi-supervised clustering and subspace clustering can be used together to create new algorithms that aim to solve problems with partially labeled data.
  - These algorithms can be created from different point of views, depending on both the employed techniques and how data labels and subspaces are used together.
  - The created algorithms lead to clustering solutions that, otherwise, cannot be such accurate or can be even unreachable.

Based on this hypothesis, the main objective of this thesis is to tackle the semi-supervised subspace clustering problem from different approaches and to develop the adequate experiments in order to validate the different approaches. In more detail, the following separated objectives can be distinguished:

1. Study of some of the most used clustering validation indices under different data conditions and clustering algorithms, as an inherent problem in all tasks related to clustering.
2. Creation of semi-supervised subspace clustering proposals from two different points of view:
   - Using available data labels for searching for subspaces firstly, before searching for clusters. This proposal assigns each instance to only one cluster (hard clustering) and is based on mapping known labels to subspaces using supervised classification techniques. Subspaces are then used to find clusters using traditional clustering techniques.
   - Using available data labels to search for subspaces and clusters at the same time in an iterative process. This proposal assigns each instance to each cluster based on a membership probability (soft clustering) and is based on integrating known labels and the search for subspaces into a model-based clustering approach.
3. Application of machine learning to neuroscience, presenting a real problem and applying the necessary techniques, including some of the proposals, to provide solutions.

As a summary, this work first attempts to throw some light about some clustering validation indices; then addresses the fundamental issues regarding the implementation of semi-supervised subspaces clustering algorithms from different points of view, highlighting their characteristics and benefits; and finally, the proposed and other machine learning algorithms are applied to a concrete neuroscience problem as an example of real and useful application. Figure 1.3 represents this summary pointing out the different objectives of this research.

1.3 Document organization

The rest of this document is divided into four thematic blocks. Each one is divided into chapters to facilitate its understanding:

- **Background:** the first block gathers the current state of the researching areas related to this thesis.
  - Chapter 2 presents an overview of machine learning, and more specifically, pattern recognition and its most common tasks. The aim of this chapter is to present a high-level scenario in which this thesis is developed. Besides, semi-supervised learning task is described, separating between semi-supervised classification and semi-supervised clustering. Specific related algorithms from both approaches are

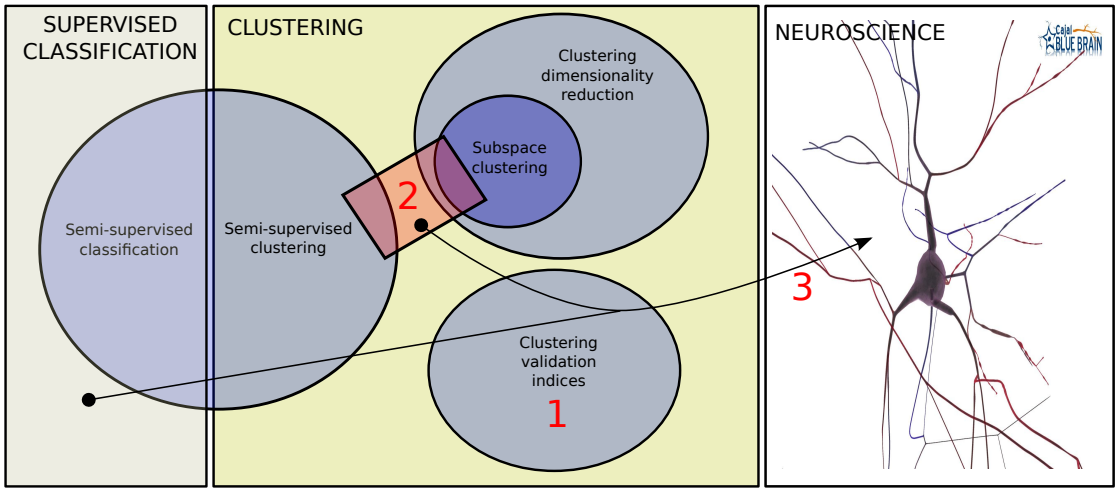


Figure 1.3: Graphical representation of this thesis objectives. Numbers match with previously presented objectives: **1.** Study of clustering validation indices. **2.** Proposals on semi-supervised subspace clustering. **3.** Application of machine learning techniques (including supervised classification techniques and some of the proposals) to neuroscience. The neuroscience figure, representing a 3-dimension reconstruction of a neuron, belongs to the Cajal Blue Brain Project (see Section 6.4).

also presented. Subspace clustering is detailed together with the state of the art of related algorithms. Finally, at the end of this chapter, some algorithms based on semi-supervised learning and subspaces, and therefore close to the proposals presented in this thesis, are presented.

- **Proposals:** the second block states the proposed approaches for semi-supervised subspace clustering together with all the experiments performed in this research.

  – Chapter 3 presents a study about some of the most widely known clustering validation indices. Validation is an inherent problem in all tasks related to clustering. This research shows the behaviour of several indices when they are used to validate clustering results obtained by using different clustering algorithms and input data characteristics.

  – Chapter 4 presents a framework based on finding subspaces from available labels using supervised techniques, and then clustering data using these subspaces and traditional clustering algorithms. A specific instantiation, using known hard clustering algorithms, is also presented and validated using synthetic and real data.

  – Chapter 5 first details model-based clustering, which is a soft clustering approach, and then, presents an extension of this kind of clustering to include both the available labels and the subspace search into the clustering process. Finally, experimental results using both synthetic and real data are also presented.

- **Applications in neuroscience:** the third block presents the neuroscience domain together with real applications related to this thesis.

  - Chapter 6 introduces basic concepts and history about neuroscience as well as some projects related to this thesis that combine interdisciplinary attempts to advance in neuroscience domain research. A specific problem, the neurons classification, is also presented.
  - Chapter 7 presents the first application of pattern recognition to neuroscience by separating two well-known types of brain cells using different supervised techniques. Besides, these techniques are also compared to unsupervised techniques, which are the most common used techniques in neuroscience for every classification task. This is the first step to solve the neurons classification problem presented in the previous chapter.
  - Chapter 8 presents another application of a supervised technique and the proposal introduced in Chapter 5 as the second step to solve the neurons classification problem, focused on one of the well-known neuron types previously separated. Due to the complexity of the task, a final solution is not achieved but some conclusions are drawn as a further step to reach a consensual solution.

- **Conclusions and future work:** the last part shows the conclusions of this thesis and the open lines of the proposals.
  - Chapter 9 extracts the most important conclusions obtained from the achievements of this work.
  - Chapter 10 describes the possible research lines that arise at the end of the thesis development.

Part II

BACKGROUND

Chapter 2

Pattern recognition

2.1 Introduction

Pattern recognition is a discipline based on classifying instances into some classes [246]. This discipline can be located within another one, even more general, called *machine learning*. Therefore, some definitions are provided first trying to clarify what machine learning means. A first definition can be found in [195]:

Definition "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

Another definition very related to explain the discipline and extracted from [262], is:

Definition "Things learn when they change their behavior in a way that makes them perform better in the future".

From both previous definitions, machine learning is about learning from data to correctly perform a task in the future. Machine learning tasks are considered pattern recognition tasks when learning from data entails finding patterns for data. Although there are different machine learning tasks, like reinforcement learning [243], the two most common tasks, supervised classification and unsupervised classification (or clustering), are pattern recognition tasks. These two tasks are different in both purpose and input data characteristics.

A process very close to machine learning and pattern recognition is *data mining*. Data mining can be seen as the application of machine learning to concrete data, and is a step within a larger process, called knowledge discovery in databases (KDD) [85]. KDD has several steps, from data acquisition to evaluation of obtained results. A graphical representation of KDD can be found in Figure 2.1.

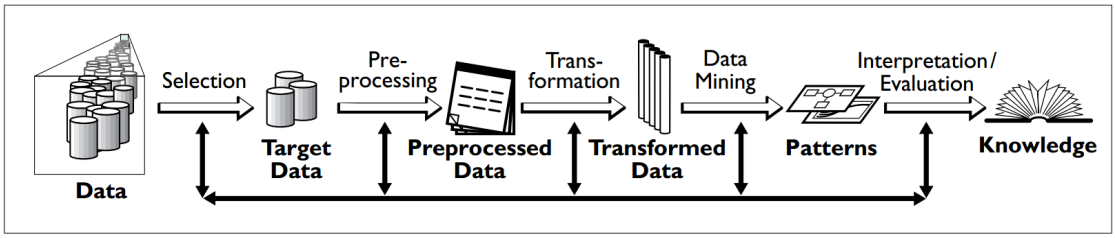


Figure 2.1: Graphical representation of the KDD process, extracted from [84].

There are some other processes, very related to KDD, and therefore to data mining, that are mixed up on some occasions. The most known example of this kind of processes is cross-industry standard process for data mining (CRISP-DM) [261]. The discussion about differences between several data analysis processes is out of the scope of this research, but based on [11], CRISP-DM can be seen as a mainly-used-in-industry implementation of KDD. In any case, data mining can be found within CRISP-DM in the "modeling" step and, therefore, the difference with KDD is not considered as significative for this research. Figure 2.2 shows the CRISP-DM graphical representation. To conclude with this discussion, note that there are other data analysis processes, like SEMMA [11], that stands for Sample, Explore, Modify, Model, Assess.

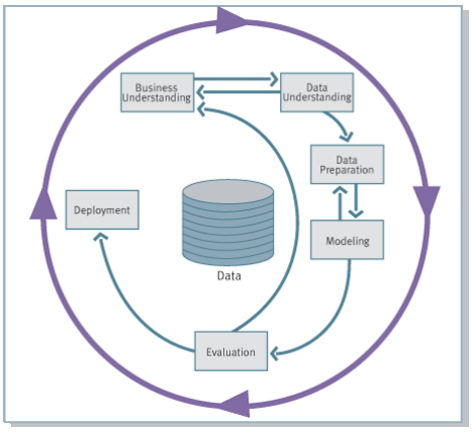


Figure 2.2: Graphical representation of the CRISP-DM process, extracted from the step-by-step data mining guide [51].

Chapter outline

After this brief introduction to some related concepts, this chapter continues giving details about supervised and unsupervised classifications. Some of the most important approaches and algorithms are also presented. The comprehension of these two tasks is necessary to understand another task, called semi-supervised learning (see Section 2.4), which is explained next together with semi-supervised clustering, which is a kind of semi-supervised learning, and is one of the main topics of this thesis. After this, subspace clustering is presented, which is the other main topic of this work. As commented later, subspace clustering is a tendency related to dimensionality reduction (see Section 2.5) and clustering. Different approaches and algorithms are also presented as state of the art. Finally, the chapter ends by presenting works in which both semi-supervised clustering and subspace clustering are used together. As commented in Chapter 1, this work is focused on this kind of approaches.

This thesis is focused on continuous data, therefore the background, related work, presented proposals, and the real application, also focus on this kind of data.

2.2 Supervised classification

Imagine you are a bank director who has bank loans documentation about your clients from the last five years. There are different features for each client within the documentation, and, as additional information, you know the current state of the loan, i.e., whether the client has fulfilled with all the payments (safe client) or whether the client has had some delays or even non-payment (risky client) during that period of time. Then, a new client, who shares the same features with all your previous clients, is asking you for a loan and you are not sure about to grant it, depending on whether you would classify your client either as a safe or a risky client. Supervised classification may help you to make a decision.

The above situation is one of the most used examples to explain what supervised classification is. Supervised classification is a two-steps predictive task: first of all, a model is built based on some input data (also called training data), and this model is then used to make predictions about future data (also called test data). The training data must be characterized using pairs of descriptive features and a class label variable (also called class or merely label), whereas the test data are characterized using only the descriptive features. The aim is to make predictions about the test data classes using a model built from the training data.

A formal definition is, let the training set $\mathcal{X} = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(N)}, c^{(N)})\}$ be a set of instances described by a tuple of a vector of descriptive features in a space of dimension F, that is, $\mathbf{x}^{(i)} \in \Re^F$, and a label from a class variable, $c^{(i)} \in \{1, \ldots, C\}$, with $i \in \{1, \ldots, N\}$, a supervised classification algorithm builds a classification model, learned from $\mathcal{X}$, which will be used to assign class labels to new instances, $\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N+M)}\}$. Hence, for a single new instance, the model can be seen as a function:

$$\gamma : \mathbf{x}^{(N+1)} \rightarrow \{1, \ldots, C\} \tag{2.1}$$

There are many different approaches to perform supervised classification, and, within these approaches, there are also many specific supervised classification algorithms. The categorization for the different supervised classification approaches depends on the source in literature. One classification attempt can be found in [119], where Han and Kamber divided the supervised approaches into classification by decision tree induction, Bayesian classification, rule-based classification, classification by backpropagation, support vector machines, associative classification, lazy learners, and others, like genetic algorithms (GAs) or fuzzy approaches. On the other hand, Kotsiantis et al. [157] reduced this categorization to logic-based algorithms, perceptron-based techniques, and statistical learning algorithms, in their supervised classification survey.

Although supervised classification is out of the main scope of this thesis, some supervised classification algorithms are used throughout the research. Therefore, the different paradigms of these used algorithms, together with an explanation of each algorithm, are presented next, as an example of some of the most known supervised classification approaches. This presentation does not pretend to cover all the supervised classification set, and other important paradigms, like support vector machine (SVM) [250] for instance, are not presented because

they are not used in this work.

2.2.1 Supervised classification approaches

Bayesian classifiers

The goal of algorithms from this approach is to predict class membership probabilities [76]. Therefore, and using prior probabilities according to Bayes' theorem, the assigned class to each new instance is the most probable a posteriori class:

$$\arg\max_c p(c|\mathbf{x}) = \arg\max_c p(c)p(\mathbf{x}|c). \tag{2.2}$$

The most known algorithm, and also the simplest, is naïve Bayes (NB) [194]. The maximum a posteriori assignment to the class label is based on obtaining the conditional probability density function for each feature given the value of the class variable, assuming conditional independence of the descriptive features given the class (see Figure 2.3), that is

$$\arg\max_c p(c) \prod_{j=1}^{F} p(x_j|c). \tag{2.3}$$

Although the conditional independence assumption does not come true in many practical scenarios, NB has obtained competitive performance when compared with other more complicated approaches. Nevertheless, the independence assumption is alleviated in other Bayesian classifiers, as seminaïve Bayes [156, 209] and tree augmented naïve Bayes (TAN) [94].

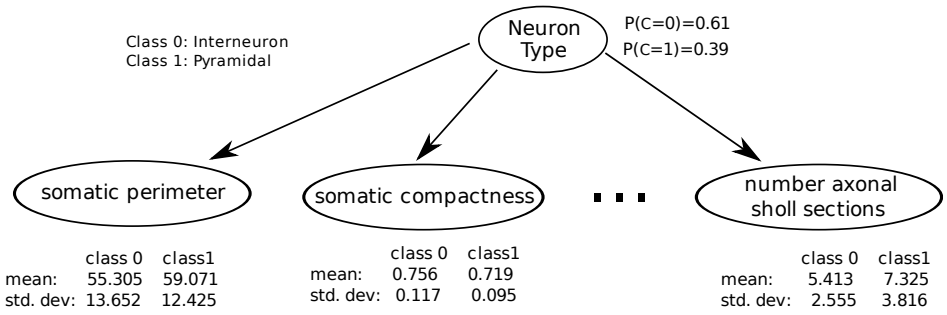


Figure 2.3: Part of a real NB graphical representation classifying neuronal data. The conditional independence assumption between features given the class variable is represented with no connections between them.

Classification trees

This approach [40] aims to build a tree structure where each node is a question related to some predictive feature. Depending on the answer to node questions, i.e., predictive feature values, new branches connect to other nodes. This structure is repeated until a leaf node is reached, where a class label is held as pattern response. Therefore, when a new instance to be classified appears, the tree structure will be covered following the proper branches depending on feature values until a class label is found in a leaf node.

C4.5 [215] is one of the most known classification trees algorithms. It builds a decision tree from the training data using recursive partitioning of the space representing the predictive features and based on the information gain ratio. A graphical representation of a real C4.5 output can be seen in Figure 2.4.

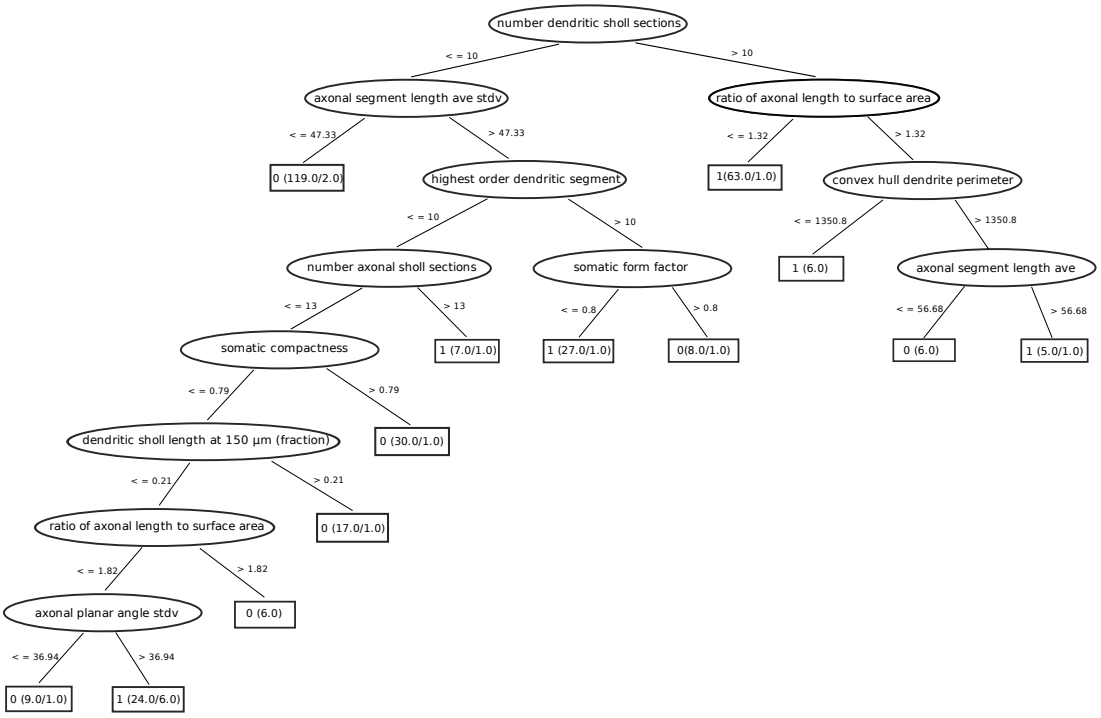


Figure 2.4: Example of a real C4.5 output representation classifying neuronal data.

Lazy algorithms

This kind of algorithms does not provide an explicit model as the other paradigms. Therefore, the whole training data set must be stored until a new instance to be classified appears, and a class label is then assigned to that instance depending on the labels of the most similar instances on the stored data.

K-nearest neighbors (K-nn) [56] is the typical example of lazy classifier. In this algorithm, a new instance is compared to the available training data according to some distance metric. The class label is then assigned, as mentioned before, depending on the class labels for the K closest training instances.

Neural networks

This approach [188] is based on weight assignments to connections between input/output units, which are often organised in different layers. The weights are adjusted during the learning phase to predict labels for new instances. Each unit is known as a "neuron", due to an analogy to simulate the structure and behavior of biological neuronal networks.

The simplest algorithm is called perceptron [228] and is able to distinguish labels in a binary classification problem by using a threshold activation function and a linear discrimination function. The perceptron structure has a single layer, but for more complex problems, a perceptron with several hidden layers can be used, creating a multilayer perceptron (MLP) [231].

Statistical theory

Logistic regression (LR) [131, 152], from statistical theory, is an algorithm that is used to predict the class of new instances in a binary classification problem by using a linear function of the predictive features as

$$p(c = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{F} \beta_j x_j)}}, \tag{2.4}$$

where $\beta_0, \ldots, \beta_F$ are the parameters of the model. The estimation of these parameters is based on the maximum likelihood estimation method. This approach can be also used for multivalued classification problems using other functions.

2.2.2 Validation

Validation of results obtained by supervised classification algorithms is relatively straightforward due to the presence of class labels, which are considered as the *ground truth*. Based on the class labels, the *accuracy* of an algorithm is the percentage of instances that are correctly classified by the algorithm. An instance is considered as correctly classified by an algorithm when the predicted class is the same than the class that was known beforehand.

Another important tool to validate results obtained by supervised classification algorithms is the *confusion matrix*. A single instance, in a binary classification problem with positive/negative classes, is considered true positive (TP) if it belongs to positive class and is correctly classified, true negative (TN) if it belongs to negative class and is correctly classified, false positive (FP) if it belongs to negative class and is classified as positive, and finally, false negative (FN) if it belongs to negative class and is classified as positive, see Table 2.1.

		Predicted class	
		positive	negative
True class	positive	TP	FN
	negative	FP	TN

Table 2.1: Confusion matrix for a binary classification problem.

Representing the total number of instances in each case with TP, TN, FP, and FN, and being N the total number of instances, several measures can be directly obtained using these values from the confusion matrix:

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.5}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{2.6}$$

$$\text{error rate} = \frac{\text{FN} + \text{FP}}{N}. \tag{2.7}$$

From information retrieval [249], and also statistical classifications, two other measures can be directly extracted from the confusion matrix:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.8}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.9}$$

Note that recall matches sensitivity. Using these terms, the previously presented accuracy can be represented as

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{N} \times 100. \tag{2.10}$$

Besides defining performance measures, it is necessary to define the method to estimate these measures, depending on how available labeled data are managed. There are several approaches:

- Resubstitution. This is the simplest estimation method. In it, the same training data set used to build the model is also used to validate it. This method is too optimistic, with a high bias to the specific used data. Therefore, resubstitution cannot be considered a honest method for estimating any performance measure of a classifier.

- Hold-out. The available labeled data are split into training and test data. The training data is then used to build a model, which is evaluated using the test data. The previous optimism disappears using this method, but its main disadvantage is that a data subset (the test data) is not used to build the model, which cannot be desirable if the available data sample size is not very high.

- K-fold cross-validation [242]. The possible lack of knowledge for building a model by using hold-out is avoided in cross-validation by randomly partitioning the available data into K mutually exclusive folds. These folds are usually balanced, i.e., there are approximately the same number of instances in each fold. The building and testing processes are then repeated K times, reserving a different fold for testing in each run. The final quality measure is calculated by averaging all runs and the final model learnt from all the original labeled instances.

- Leave-one-out cross-validation. This can be seen as a concrete K-fold cross-validation case in which $K = N$. It means that the learning process is repeated N times, using $N - 1$ instances to learn, and using a single instance to test each time.

- Bootstrap [80]. This is a general sampling approach based on uniformly selecting instances with replacement. There are several bootstrap methods, being 0.632 bootstrap [81] one of the most commonly used. This method creates a new data set of N instances by sampling the available data set N times with replacement. Some instances might be selected more than once during the sampling process, and therefore, some instances in the original data set will not be selected. The no selected instances will be the test data. The name of this method is related to the probability of an instance to be selected or not: an instance has $1/N$ probability of being selected and a $1 - 1/N$ of not being selected each time. Thus, the probability for an instance of not being selected is

$$(1 - \frac{1}{N})^N \approx e^{-1} = 0.368.$$

  Assuming then a data set large enough, the test and the training data sets generated will contain about 36.8% and 63.2% of the instances, respectively. The name of the method comes from the probability of the generated training data set. Results obtained with this method may be biased due to the training data set size. For this reason, a possible error rate $\hat{\epsilon}$ is estimated as

$$\hat{\epsilon} = (0.368 \times \hat{\epsilon}_{train}) + (0.632 \times \hat{\epsilon}_{test}).$$

2.3 Unsupervised classification

Imagine now you are a mobile phone company director; you are interested in offering different interesting products to your clients depending on their charges, phone calls time, and many other parameters. Therefore, you would like to partition your clients into different groups, joining those that might have a similar profile, to personalize the offered products for each group. Unsupervised classification (or clustering) may help you to get success in this task.

Clustering is a descriptive task that aims at obtaining a data division by grouping instances based on the input data and some similarity measure. In this case, the input data is only characterized using descriptive features and, as commented above, the aim is, based on some similarity measure, partition the data into different groups (often called clusters).

A formal definition of clustering is, let the data set $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ be a set of instances described by a vector of descriptive features in a space of dimension F, that is, $\mathbf{x}^{(i)} \in \Re^F, \forall i \in \{1, \ldots, N\}$, the goal is then to assign a cluster label $c^{(i)}$ to each instance, with $c^{(i)} \in \{1, \ldots, K\}$, based on some similarity measure with the other instances, and K being the number of clusters in the data. Note that the cluster assignment can be considered similar to the class label assignment in supervised classification. One of the main differences is that supervised classification assigns a class label from a set of known labels, whereas clustering finds previously unknown labels (clusters). The final number of clusters, K, is often unknown and must be estimated.

The categorization of the different clustering approaches may depend on several criteria.

In [137], one of the most cited clustering surveys, Jain et al. distinguished between agglomerative and divisive (depending on the algorithmic structure), monothetic and polythetic (depending on how features are used), hard and fuzzy (depending on the kind of instance memberships; hard clustering is also called crisp clustering, whereas fuzzy clustering is also called soft clustering), deterministic and stochastic (depending on optimization techniques), and, finally, incremental and non-incremental (depending on how instances are used). The same authors separated, at the very top level of a possible hierarchy, between hierarchical and partitional approaches. The former produces a sequence of partitions and the latter directly partitions data into a fixed number of clusters. This rough separation was also accepted in [83]. Xu and Wunsch II [269] also followed this frame in their survey of clustering, but they also separated partitional approaches into squared error-based algorithms and mixture models algorithms. Besides, these authors also introduced other techniques, that can be used for hierarchical and partitional clustering, including graph theory [136, 147], combinatorial search techniques (like evolutionary algorithms (EAs) [87] or GAs [104, 129]), fuzzy set theory [101, 130, 278], neural networks [154, 155, 204], and kernel techniques [201, 233, 251]. The mixture models approach, also called model-based clustering, is considered as an independent approach by Han and Kamber [119]. Other approaches, such as density-based algorithms or grid-based methods (see Section 2.5) are also included into the clustering approach categorization in [119]. Density-based methods, with density-based spatial clustering of applications with noise (DBSCAN) [82] as the most representative algorithm, are also included within the partitional approach in [25]. Finally, another relatively recent approach is called affinity propagation [92]. This method defines *exemplars* as the most representative examples of a data set. When the *exemplars* are refined, the set of clusters gradually emerges. The main advantages of this algorithm are that it does not need the number of clusters as input parameter and is faster than other methods. From all these different categorizations, it is evident that there are many clustering approaches in the literature.

Hierarchical, partitional, and model-based clustering are somehow used throughout this thesis. Due to this, the remainder of this chapter is focused on the introduction and overview of these approaches. Before that, and as an important choice in some kinds of clustering, like hierarchical or some partitional, different metrics that can be used to calculate the similarity between two groups of instances are presented.

Clustering metrics

The metric plays a key role in some clustering approaches, like hierarchical or partitional, since the proximity between two instances can be completely different depending on the used metric. When working with continuous attributes, as in this thesis, the proximity between two instances is typically quantified by dissimilarity measures [83]. These measures can be broadly divided into distance measures and correlation-type measures.

One of the most commonly used distance measures is called Euclidean distance. For instance, the distance (d) between two consecutive instances $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(i+1)}$ is calculated

with the Euclidean distance as

$$d_{(\mathbf{x}^{(i)},\mathbf{x}^{(i+1)})} = \sqrt{\sum_{j=1}^{F}(x_j^{(i)} - x_j^{(i+1)})^2}. \tag{2.11}$$

The Euclidean distance is formally known as the l_2 norm because it is a concrete case of the general Minkowski distance (or l_r):

$$d_{(\mathbf{x}^{(i)},\mathbf{x}^{(i+1)})} = \sqrt[r]{\sum_{j=1}^{F} w_j^r(x_j^{(i)} - x_j^{(i+1)})^r}, \tag{2.12}$$

where w_j is a possible importance weight for feature j, and r the distance norm. Manhattan distance (or l_1) is another measure following this structure, with $r = 1$.

There are other different measures, such as Mahalanobis distance or Pearson's correlation [211], based on correlations between features. The definition of the Pearson's correlation is

$$\delta_{(\mathbf{x}^{(i)},\mathbf{x}^{(i+1)})} = \frac{1 - \phi_{(\mathbf{x}^{(i)},\mathbf{x}^{(i+1)})}}{2}, \tag{2.13}$$

with

$$\phi_{(\mathbf{x}^{(i)},\mathbf{x}^{(i+1)})} = \frac{\sum_{j=1}^{F} w_j(x_j^{(i)} - \overline{\mathbf{x}}^{(i)})(x_j^{(i+1)} - \overline{\mathbf{x}}^{(i+1)})}{(\sum_{j=1}^{F} w_j(x_j^{(i)} - \overline{\mathbf{x}}^{(i)})^2 \sum_{j=1}^{F} w_j(x_j^{(i+1)} - \overline{\mathbf{x}}^{(i+1)})^2)^{1/2}}, \tag{2.14}$$

where

$$\overline{\mathbf{x}}^{(i)} = \frac{\sum_{j=1}^{F} w_j x_j^{(i)}}{\sum_{j=1}^{F} w_j}. \tag{2.15}$$

The presented examples are some of the most commonly used dissimilarity measures. For further details about them and many other measures, see [72].

2.3.1 Unsupervised classification approaches

Hierarchical clustering

Hierarchical clustering aims at grouping data into a hierarchy structure. A proximity matrix among the instances is built based on some dissimilarity measure, and nested partitions are then obtained according to it. A review of hierarchical clustering can be found in [108]. There are two paradigms of hierarchical clustering algorithms:

1. Divisive clustering. This is a top-down clustering. All instances start grouped together into the same cluster, and this cluster is then split, according to the largest between-group dissimilarity, obtaining two different clusters. In each step of the process, a cluster is subdivided into two smaller clusters until each instance forms a single cluster.

2. Agglomerative clustering. This is a bottom-up clustering. At the beginning of the process, each instance forms a single cluster. Two clusters are merged at each step according to the between-group similarity. Again, this is an iterative process that continues until all instances are grouped together into the same cluster.

A limitation of hierarchical clustering is that divisions in the divisive, and mergers in the agglomerative paradigm, cannot be undone once made.

The output hierarchy is often plotted in a structure called *dendrogram* (see an example in Figure 2.5). This kind of plot is easily readable since it represents not only the joining between instances but also the proximity among them. Based on this dendrogram, and choosing a cutting point on it, the user is able to select the final output clustering. Therefore the number of final clusters is not an input parameter in hierarchical clustering, but it has to be chosen at the end of the process if a unique clustering solution is required.

Although the number of clusters is not an input parameter, some choices must be made in order to decide whether a cluster must be split (divisive paradigm) or whether two clusters must be joined (agglomerative paradigm). The user must choose a metric (commented above) and a linkage criterion.

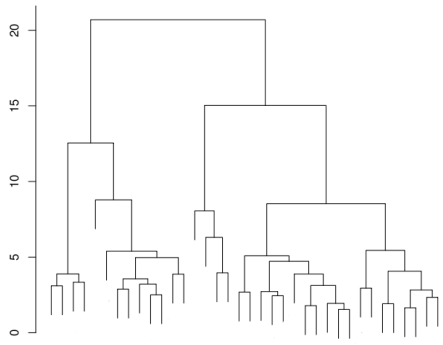


Figure 2.5: Dendrogram representing an output obtained by running an agglomerative hierarchical clustering.

Linkage criteria Metrics presented in this section above are valid to obtain the dissimilarity between two instances, but when a cluster is formed by several instances, a linkage criterion is necessary. Therefore, a linkage criterion is the definition of distance between either two clusters, or an instance and a cluster.

A complete description of each criterion can be found in [83], but a brief introduction to some of the most commonly used linkage criteria is presented:

- Single linkage [238]. This is the minimum distance criterion, i.e., the distance between two clusters is obtained by calculating the distance between the two closest instances in each cluster.

- Complete linkage [240]. This is the maximum distance criterion, i.e., the distance between two clusters is obtained by calculating the distance between the two most remote instances in each cluster.

- Average linkage [239]. The distance between two clusters is the average of all distances between pairs of instances of each cluster.

- Ward's method [259]. The distance between two clusters is calculated depending on the

variance between the clusters. Hence, the minimum distance is obtained by minimizing the within-cluster variance.

Partitional clustering

Partitional clustering aims to partition the data, based on some dissimilarity measure, into a pre-fixed number of clusters.Therefore, a single data partition is obtained. This is the major difference between hierarchical and partitional clustering.

The most representative partitional clustering algorithm is K-means [88, 175, 179, 241]. As mentioned above, K-means needs K, the number of clusters, as input parameter, and aims to cluster the data set $\mathcal{X}$ into K clusters. The algorithm proceeds as follows. First, the K clusters are initialized (there are many variants for this initialization, a comparison of four of them can be found in [210]), obtaining K centroids, which represent the K cluster centers, being μ_k the centroid of cluster C_k. Each instance $\mathbf{x}^{(i)}$ is assigned to a cluster by minimizing the distance between the instance and cluster centroids. Once each instance is assigned to a cluster, the cluster centroids are recalculated based on those assignments. After the new centroids are calculated, the instances are again reallocated in the clusters. This is an iterative process that converges when cluster centroids do not suffer any changes from an iteration to another. At the end, the aim is to find a partition P such that the squared error between the centroids (μ) and the instances ($\mathbf{x}$) is minimized:

$$J(P) = \sum_{k=1}^{K} \sum_{\mathbf{x}^{(i)} \in C_k} \|\mathbf{x}^{(i)} - \mu_k\|^2. \tag{2.16}$$

An example of K-means execution, extracted from [29], clustering the data set called *old-faithful* (representing measurements of the eruption of the Old Faithful geyser at Yellowstone National Park in the United States) can be found in Figure 2.6. Instances are represented in a re-scaled 2-dimensional Euclidean space, the number of fixed clusters is two ($K = 2$), and cluster centroids, μ_1 and μ_2, are represented by a blue and a red cross, respectively. The initialization of the algorithm is shown in (a). Each instance is assigned to a cluster in (b) depending on the distance to each cluster centroid. After that, cluster centroids are recalculated in (c) according to the instance assignments. This process is repeated until the convergence criterion is reached, i.e., cluster centroids do not suffer any changes.

K-means was created more than fifty years ago, but its importance was still pointed in [135] recently. Some extensions related to K-means are K-medoids [148], representing the clusters using the median, and fuzzy C-means [26, 78], which is the K-means variant for soft clustering.

Model-based clustering

This clustering approach is based on the assumption that data were generated using several probability distributions. Thus, in clustering terms, each probability distribution can be seen as a cluster (also called component in this approach), and instances in different clusters were

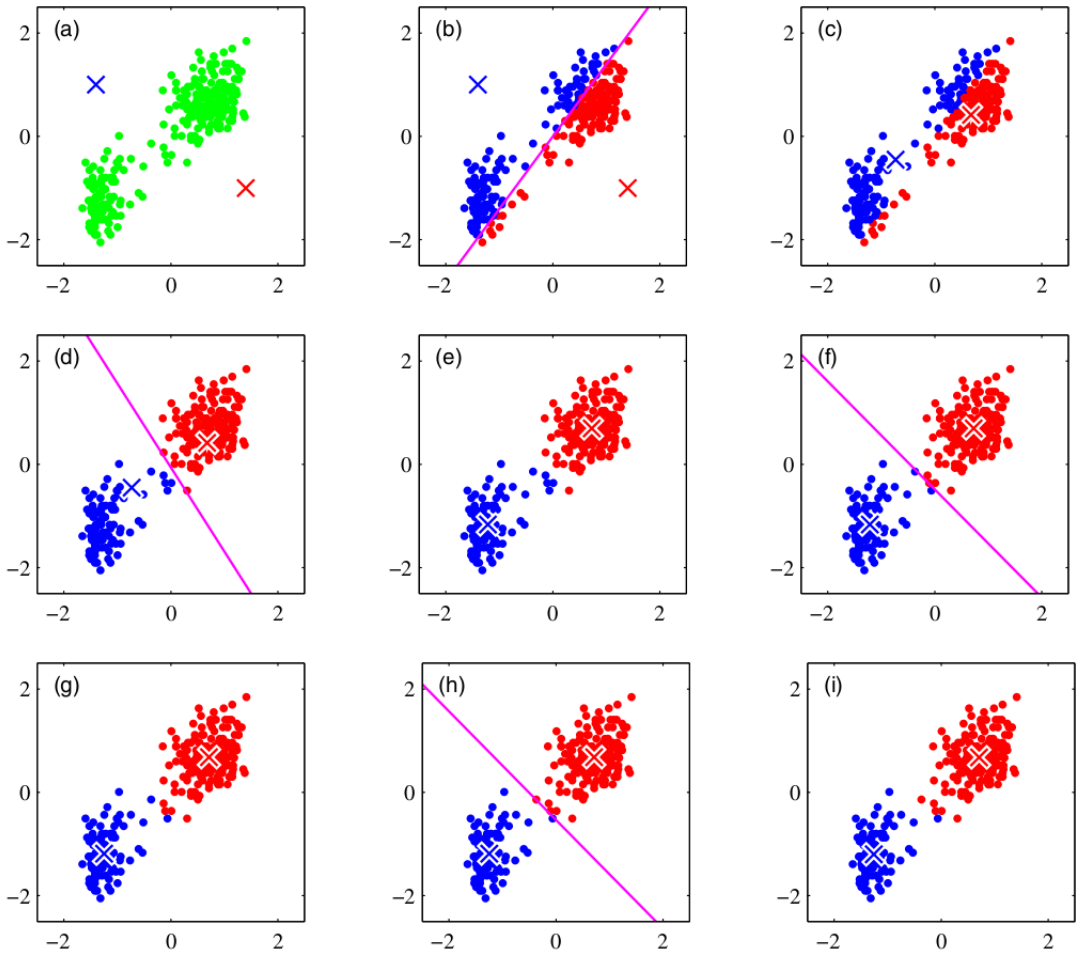


Figure 2.6: Example, taken from [29], of K-means execution clustering the *oldfaithful* data set. Initialization is shown in (a), cluster assignments in (b), and cluster centroids recalculation in (c). The process of assignments and recalculations is repeated until the convergence criterion is reached.

generated by different probability distributions. The mixture of all the distributions, called finite mixture model [189], can therefore model all the data as a whole. Although other distributions, like Poisson [165] or skew-normal [168], can be used to model each mixture component, by far, the most popular mixture model is formed by Gaussian components [190].

Using this approach, the clustering problem becomes a mixture parameters estimation problem. Once the parameters are estimated, they can be used to calculate the posterior probabilities of each instance and distribution, i.e., the membership of each instance to each cluster (soft clustering). The parameter estimation is often performed using the expectation-maximization (EM) algorithm [70]. This is an iterative method that estimates the model parameters by finding maximum likelihood estimates. Although the EM algorithm is also used in Chapter 3, a detailed explanation of the basic algorithm can be found in Section 5.2, since the algorithm is extended throughout Chapter 5 in one of the proposals of this thesis. As

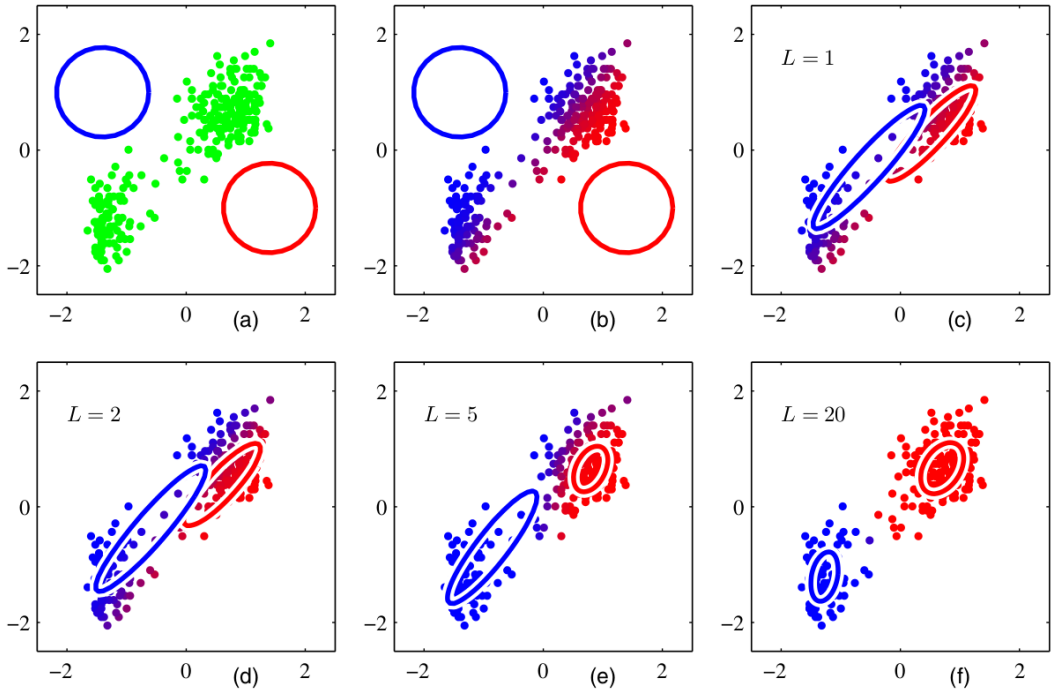


Figure 2.7: Example, taken from [29], of model-based clustering execution, with the *oldfaithful* data set using the EM algorithm and two components to model the mixture. Components are initialized in (a). Soft cluster assignment (E-step) is represented in (b), and finally, cluster parameter recalculation (M-step) is shown in (c), finishing the first iteration ($L = 1$) of the algorithm. Iterations 2, 5, and 20 are also shown in (d), (e), and (f), respectively.

an introduction, the process of the EM algorithm is very similar to the above presented process of the K-means algorithm. Both of them begin with cluster initializations (parameters in EM and centroids in K-means). After the initialization, the iterative process begins and instances are assigned to different clusters (soft clustering in E-step of EM and hard clustering in K-means). Clusters are then recalculated depending on the previous assignments (parameter estimation in M-step of EM and centroids in K-means). Finally, if the convergence criterion has not been reached, the process iterates again. All this process applying the EM algorithm is shown in Figure 2.7, using again the re- scaled *oldfaithful* data set, and two components to model the mixture. This figure, which is also extracted from [29], shows how the EM process is very similar to K-means, starting with an initialization step in (a). The soft cluster assignment (E-step) is performed in (b), where the different colour intensities show the membership of each instance to each component. The first EM iteration ($L = 1$) finishes with the cluster parameter recalculation (M-step) in (c). Results at the end of iterations 2, 5, and 20 are shown in (d), (e), and (f), respectively.

The mixture parameters initialization is a critical decision for finding the best local maximizer, or even the global maximizer. See [86, 181] for complete overviews of many different initialization procedures. These can be divided into deterministic and stochastic procedures. The former are less demanding, but algorithms from this approach are not able to propose

starting values different from the initially chosen; an example from this approach can be seen in [90]. Stochastic procedures can restart the starting values if the initially ones were not properly selected, but this process leads to more demanding algorithms, like [27]. Some comparisons of several initialization procedures can be found in [146, 182]. Nevertheless, parameters initialization is not a resolved topic in model-based clustering, because no method uniformly outperforms the others. Due to this, new proposals appeared recently, like [191], where Melnykov and Melnykov proposed an initialization depending on high density instances areas, meanwhile using a truncated normal distribution, and [271], where Yang et al. proposed a new schema using all instances as initials to solve the problem of choosing initial values.

The number of components of the mixture is often unknown. This number can be considered as an input parameter of model-based clustering (similar to the number of partitions in K-means) that must be estimated. Different available approaches to address this problem are overviewed in [189]. Most of these approaches are based on choosing the number of components depending on the optimization of the likelihood value augmented by some penalty function according to the model complexity. Therefore, several number of components are tried to build different models. The model with the most optimized value is then selected. Some information-based criteria are usually chosen as value to be optimized, two of the most common are Akaike's information criterion (AIC) [6] and Bayesian information criterion (BIC), also known as Schwartz criterion [234].

2.3.2 Validation

Clustering solutions are usually much more difficult to evaluate than supervised classification solutions, because there is no *ground truth*, i.e., class labels. A clustering solution quality is closely related to domain expert opinion in many situations. In spite of this, there are lots of clustering quality indices (CQIs) that try to assess the quality of clustering solutions. To do this, indices tend to rate compact and isolated clusters highly.

Regarding the classification of CQIs, Yeung et al. [272] indicated that there are two groups of CQIs: internal and external. These types of indices match with the equally called internal and external criteria, which are traditionally used for clustering validation when appropriate. Meanwhile, Jain and Dubes [136] and Halkidi et al. [117] discussed the existence of a third criterion, called relative criterion. The relative criterion concept is based on creating several clustering partitions depending on an assumed criterion, and choosing then the clustering partition which best satisfies such criterion. A typical example of criterion is the number of clusters: the relative criterion for clustering validation consists of running the clustering algorithm from a minimum to a maximum number of clusters and the best clustering partition according to some index is chosen as the best partition. The typical used indices in relative criteria are the internal indices, which are presented next.

Internal indices

Internal indices do not require any knowledge about the *ground truth*. A clustering partition quality, using internal CQIs, is assessed by evaluating the partition based on distance or dissimilarity measures. The problem with this approach is that results may be very biased depending on how the partition was built, since the quality may be measured with different criteria than those used to build the partition, which can lead to incorrect validations. Therefore, CQIs suffer from biases not only regarding the data (shape or number of clusters for instance), but also regarding to the clustering algorithm used to obtain the partition [121].

The partition quality is often used as stopping rule for finding the correct number of clusters hidden in a data set. Following this approach, Milligan and Cooper [193] ranked 30 internal CQIs on an extensive battery of data sets without impurities (clear cluster structures). These data sets had different configurations by varying the number of features and instance distribution density levels (instances were not equally distributed in each hidden group). Hierarchical clustering was used as clustering approach for grouping the data. This comparison is still considered as one of the main references for clustering validation even though the work was developed more than 25 years ago. The importance of this work is exemplified by current works dealing with the same issue, such as the research of Vendramin et al. [252], where authors used the same type of data sets as in [193] but changed the evaluation approach by introducing another type of CQIs for validation. Besides these, there are some other references in the literature that attempted to evaluate internal CQIs using different clustering algorithms and kinds of data, like [73, 187]. Another very recent paper comparing internal CQIs is [116]. This paper attempted to obtain a standard methodology for evaluating results obtained with hierarchical clustering and using different indices. They considered that it is necessary not only to get the correct number of clusters but also to obtain the partition that best fits the original data.

The problem with data used in the works commented above is that they are not typical cases of real domains because of the low dimensionality and the clear separation and cohesion of the clusters. This is an utopian scenario in real problems, since there are usually irrelevant or noisy features, or even no cluster structures in data. This is the main motivation for a research made in this thesis (see Chapter 3), whose conclusions are taken into account during the first proposal, presented in Chapter 4.

The internal CQIs used throughout this thesis are some of the most known and widely used in the literature. These indices are:

Silhouette [229] is calculated for an instance $\mathbf{x}^{(i)}$ as follows:

$$\text{Silhouette}(\mathbf{x}^{(i)}) = \frac{b(\mathbf{x}^{(i)}) - a(\mathbf{x}^{(i)})}{\max(b(\mathbf{x}^{(i)}), a(\mathbf{x}^{(i)}))}, \tag{2.17}$$

where $a(\mathbf{x}^{(i)})$ is the average dissimilarity between instance $\mathbf{x}^{(i)}$ and all other points in the cluster where $\mathbf{x}^{(i)}$ belongs, and $b(\mathbf{x}^{(i)})$ is the minimum average dissimilarity to instances of

each different clusters. The average of all output values is the average Silhouette, which is the final result and is in the $[-1, 1]$ range. A high value indicates good quality clusters.

Calinski [42] consists of finding well isolated clusters and is based on two measures that evaluate separation, with the between-cluster sum of squares (BSS), and cohesion, with the within-cluster sum of squares (WSS):

$$\text{Calinski} = \frac{\text{BSS}_K(K-1)}{\text{WSS}_K(N-K)}, \tag{2.18}$$

where K is the number of clusters and N is the total number of instances. The aim is to find a value of K that maximizes the index. A high value indicates isolated and unified clusters.

C-index [14] is defined as:

$$\text{C-index} = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{2.19}$$

where S is the sum of distances over all pairs of instances from the same cluster. If P is the number of those pairs of instances, S_{max} and S_{min} are the sum of the P largest and smallest distances, respectively, considering all the pairs of instances. Again, this index should be minimized and is confined to the interval $[0, 1]$.

Davies-Bouldin (DB) index [65] is calculated by averaging each pair of clusters as:

$$\text{DB} = \frac{1}{K} \sum_{k=1, k\neq k'}^{K} \max \left(\frac{d_k + d_{k'}}{d(\mu_k, \mu_{k'})} \right), \tag{2.20}$$

where K is the total number of clusters, d_k and $d_{k'}$ are the average distances of all instances in each cluster to their respective cluster center μ_k and $\mu_{k'}$. Finally, $d(\mu_k, \mu_{k'})$ is the distance between cluster centers. The target value for the DB index is a small one, and it corresponds to compact and well-separated clusters.

Gamma [14], also known as Baker and Huberts index, is defined as:

$$\text{Gamma} = \frac{s(+) - s(-)}{s(+) + s(-)}, \tag{2.21}$$

where $s(+)$ is the number of consistent comparisons and $s(-)$ the number of inconsistent comparisons. Comparisons are made between all clusters pairwise and all between-clusters pairwise dissimilarities. A comparison is consistent if a within-cluster distance is less than a between-cluster distance, otherwise it is considered as inconsistent. The target value of this index is the maximum value and it is bounded by 1.

External indices

On the other hand, external validation is more accurate but not realistic in clustering. In this case, the *ground truth* must be known and the evaluation is carried out based on this knowledge. Although there are many external CQIs, some of them are equivalent [7]. This approach has a more realistic usage when the aim is to compare different clustering partitions, even during the clustering process, like during the process of ensemble clustering, as pointed by Vinh et al. [253]. These authors classified external indices to compare clustering partitions into:

- Pair counting based measures, which are based on the agreement or disagreement between pairs of instances on the partitions. Measures presented below are examples of this kind of measures.
- Set matching based measures, which are more proper measures for supervised classification, because they are based on finding matches between cluster partitions having as requirement that both partitions have the same number of clusters.
- Information theoretic based measures, such as entropy or mutual information, which are based on concepts from information theory [57], are defined using the marginal and joint distributions of instances in each clustering partition. Precision and recall, previously explained in Section 2.2.2, are examples of this kind of measures.

External CQIs used throughout this thesis are defined taking int account the next context: given a set of N instances, suppose $\mathcal{P}$ and $\mathcal{P}'$ are two different partitions of not necessarily the same size to be compared. Then, a is the number of pairs of instances that are located in the same group in $\mathcal{P}$ and in $\mathcal{P}'$; b is the number of pairs of instances located in the same group in $\mathcal{P}$, but not in $\mathcal{P}'$; c is the number of pairs of i instances located in the same group in $\mathcal{P}'$, but not in $\mathcal{P}$; and d is the number of pairs of instances located in different groups in both partitions $\mathcal{P}$ and $\mathcal{P}'$.

Rand index [225] is defined as

$$\text{Rand} = \frac{a+d}{a+b+c+d}. \tag{2.22}$$

The problem of Rand index is its value when two random partitions are compared, since it does not take a zero (minimum) value. This index must be maximized.

Adjusted Rand index (ARI) [132] was proposed to overcome the Rand index limitation concerning random partitions. This index, like Rand, must be maximized and outputs 1 when there is a perfect match between two partitions. ARI is calculated as

$$\text{ARI} = \frac{\binom{N}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{N}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}. \tag{2.23}$$

Therefore, ARI is a very demanding index because it introduces a penalization to avoid the possibility of random classification.

Gamma index [136] is

$$\text{Gamma} = \frac{ad - bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}. \tag{2.24}$$

As ARI, Gamma index considers both instances in the same group and instances in different groups in the two partitions as good decisions. Gamma results have a range $[-1, 1]$ and must be maximized.

Gower [109] is defined as

$$\text{Gower} = \frac{a+d}{a + \frac{1}{2}(b+c) + d}, \tag{2.25}$$

and attaches more importance to the good decisions than to the bad decisions. Taking this into account, and that results, between 0 and 1, must be maximized, this index thus yields very high values.

Russel index [232] is defined as

$$\text{Russel} = \frac{a}{a+b+c+d}. \tag{2.26}$$

This index only considers pairs of instances in the same group in both partitions as good decisions. This is one of the reasons why Russel index returns low values very often. The results range is $[0, 1]$.

Clustering validation is still a challenging task nowadays. Nevertheless, validation is very important not only to assess the quality of a partition after the clustering process, but also to select the number of clusters during the clustering process. Some recent research dealing with clustering validation can be found in [266] and [206], whereas Rendón et al. recently compared internal and external indices [226].

2.4 Semi-supervised learning

Due to changes in available data, new machine learning tasks are playing an important role nowadays. These changes are often related to the available data labels. Recently, with advances in data acquisition and storage, the amount of available data is not a bottleneck in many domains. However, data labels are still scarce in some domains because they are either expensive, labour intensive to gather, or merely, completely unknown. Related to this, and as previously seen in this chapter, when all available data are labeled, supervised classification is used to perform the learning task, whereas when there are no available data labels, clustering tasks can be used for finding such labels. Nevertheless, when there are only some available data labels, with the rest of data without labels, i.e., partially labeled data,

supervised classification and clustering can be used, depending on the number of available labels and with some limitations. This scenario leads to another pattern recognition task, called semi- supervised learning [49]. The obvious point regarding data labels is that they might contribute with valuable information, not only for validating, but also for improving the learning task.

Based on [285], semi-supervised learning can be separated into inductive semi-supervised learning and transductive learning. The former aims to predict labels on future test data, as in a supervised classification problem, but using both labeled and unlabeled data in the training set to build the model. On the other hand, transductive learning aims to predict the labels on the unlabeled data in the available training set. This thesis is focused on transductive learning. Besides, depending on whether the unlabeled instances can be classified according to one of the known labels or there is the possibility of discovering new previously unknown groups, semi-supervised learning can precisely referred as semi-supervised classification or semi-supervised clustering, respectively.

A formal description of this problem is let $\mathcal{X}$ be a set of N instances described by continuous features in a space of dimension F, that is, $\mathbf{x}^{(i)} \in \Re^F, \forall i \in \{1, \ldots, N\}$. Besides, the class information of some instances is available in this partially labeled data set. Thus, $\mathcal{X}$ can be divided into $\mathcal{X} = \mathcal{X}^L \cup \mathcal{X}^U$, where $\mathcal{X}^L = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(L)}, c^{(L)})\}$ is the subset of instances with an associated known class label, with $c^{(i)} \in \{1, \ldots, C\}$. On the other hand, $\mathcal{X}^U = \{\mathbf{x}^{(L+1)}, \ldots, \mathbf{x}^{(N)}\}$ are the instances with unknown labels. The aim is to estimate the labels of those instances in $\mathcal{X}^U$: thus, $\forall j \in \{L+1, \ldots, N\}$, $c^{(j)} \in \{1, \ldots, C\}$, must be estimated in semi-supervised classification; and $c^{(j)} \in \{1, \ldots, K\}$, being $K \geq C$ the final number of clusters, must be estimated in semi-supervised clustering.

An overview of semi-supervised classification and semi-supervised clustering approaches is presented next.

2.4.1 Semi-supervised classification

Semi-supervised classification is meaningful, compared to supervised classification, when the built model using both labeled and unlabeled data is better than the one built using only labeled data. It is noteworthy that, depending on the problem and the assumptions, using semi-supervised classification can lead to a decrease in the classifier accuracy. There is not much research about this topic in the literature, but an example can be found in [58].

Based on [283], there are several semi-supervised classification methods: self-training, co-training, transductive support vector machines, graph-based methods, and EM with generative mixture models. A similar enumeration can be also found in [213].

Nevertheless, all these methods can be classified into two wider approaches based on their underlying assumptions [49, 183]. The first assumption, and therefore the first group of methods, is the *manifold assumption*, which formulates that data lie on a low-dimensional manifold. This reduces the search space of the learning algorithm, obtaining some advantages like avoiding the *curse of dimensionality* (see Section 2.5). Data are often represented in a graph by algorithms based on this assumption. Thus, the above commented graph-based

methods are within this group. Some examples of algorithms, based on the manifold assumption, are label propagation [284], Markov random walks [244], graph cut [30], or low density separation [50]. Most of these are transductive algorithms.

The second assumption is called the *cluster assumption*, which formulates that data with high similarity, i.e., data in a same cluster, must share the same class label. This assumption means that instances with different class labels shall not be grouped into the same cluster. However, this assumption does not imply that every single class forms a single cluster. Several inductive semi-supervised algorithms can be found within this group. Co-training [31] is based on that each instance can be characterized by different sources. Sources can be, for instance, two different, independent, and compatible feature sets [31], or two diverse supervised models [105] that label the instances depending on confidence intervals and noise control mechanisms. A classifier from each source is separately learnt using the labeled instances and, depending on the confident of each classifier, predictions are used to iteratively obtain new labeled data. This approach is a kind of ensemble or hybrid method, such as [24]. Another example of algorithm based on the cluster assumption, and similar to the last approach presented, is the Nigam et al. research [203], based on a generative approach (using mixtures of distributions and EM algorithm). The aim is to estimate the parameters of the distributions, first learning from the labeled instances and improving then those estimations in repeated iterations, based on the likelihood and all the instances, in [192] and [282]. Finally, other examples of algorithms based on the cluster assumption are self-training [227] and transductive support vector machines [140]. Chawla and Karakoulas [52] presented an empirical study of several of these techniques.

Finally, some other examples of semi-supervised classification, specialized on some supervised techniques, can be found in [46], based on neural networks, and in [258], based on nearest neighbors classifiers. Note also that some authors, such as Bouchachia [34], distinguished another type of semi-supervised classification. This different approach is based on obtaining labels of the unlabeled instances, using some information extracted from the available labels. Once all the data set is labeled, a supervised algorithm can then be used to build a predictive model. This approach is called pre-labeling, and some examples can be found in related work from [34].

2.4.2 Semi-supervised clustering

The available information in semi-supervised clustering is often called constraints, which leads to a different name for this task: clustering with constraints or constrained clustering [20]. There are different types of clustering constraints. Many of them are out of the scope of this thesis, but some examples are:

- Related to the number of instances in each cluster. This is when information about the number of instances in each cluster is known, or even when the search for clusters is biased to obtain balanced clusters (similar number of instances in each cluster). Some examples can be found in [39, 96, 281].

- Negative information. This is when some sets of clusters are available, from different executions of different clustering algorithms for instance, and a new clustering solution must then be found, being different enough from the available ones [107].

- Pairwise constraints. These are the most common types of clustering constraints. They are also called instance-level constraints. This kind of constraints consists of statements about pairs of instances. There are two types [255]: *must-link* and *cannot-link constraints*. A must-link constraint between two instances means that these two instances should be grouped into the same cluster. On the other hand, a cannot-link constraint between two instances indicates that they should be grouped into different clusters. When data labels are available, like in partially labeled data mentioned above, they can be directly translated into pairwise constraints. If two instances have the same class label, a must-link constraint can be inferred. In a similar way, when two instances have different class labels, there is a cannot-link constraint between them. On the other hand, when pairwise constraints are available instead of class labels, groups of instances that belong to the same cluster can be obtained from the must-link constraints [111]. One of the first surveys in clustering with instance-level constraints can be found in [60].

The last approach is the closest one to this thesis, but before presenting different approaches to clustering with pairwise constraints, the next question should be answered: Do constraints benefit the clustering performance? It is similar to the first paragraph of the previous section, where the same question about using semi-supervised classification instead of supervised classification was commented. There are different works trying to answer this question. The concept of *feasibility*, i.e., whether a clustering algorithm is able to find a clustering solution that satisfies all available constraints, can be found in [62]. Wagstaff et al. [254] presented a study about two constraints properties: *inconsistency* and *incoherence*. The former is the amount of conflict between constraints and the clustering algorithm (very close to feasibility), and the latter is the amount of internal conflict between constraints, given a distance metric. Wagstaff et al. recommend to select those constraints that minimize both inconsistency and incoherence. Finally, a related work dealing with the possible decrease of performance when using constraints in clustering, even when constraints are completely reliable, can be found in [64].

Traditionally, as compared in [18], constrained clustering algorithms have been based on two approaches: search-based and similarity-based methods. The first approach is based on the penalization of an objective function for evaluating clusterings depending on the available knowledge. On the other hand, similarity-based methods train a metric using that knowledge. From these two approaches, there is a new trend called hybrid methods by some authors [47, 230]. Hybrid methods integrate both search-based and similarity-based at the same time.

According to this small categorization, a presentation of some of the most known semi-supervised clustering algorithms follows. Besides, a summary of all these algorithms, depend-

ing on the category and the intrinsic basic clustering approach used, can be found in Table 2.2.

	Semi-supervised clustering category		
Clustering approach	Search-based	Similarity-based	Hybrid
Hierarchical	[63]		[151]
Partitional	[16, 17, 256]	[18, 44, 268]	[19, 28, 273]
Model-based		[55]	[192, 162, 177, 236]
Others	[35, 69, 230, 267]	[12, 13]	

Table 2.2: Summary of semi-supervised clustering algorithms regarding how the available information is used and the intrinsic basic clustering approach.

Davidson and Ravi [63] presented the first work that modified hierarchical clustering algorithms so as to satisfy all the available constraints. This work, which is an extension of [61], highlighted how constraints not only improve the created dendrogram but also the efficiency of the algorithm.

Klein et al. [151] also used a hierarchical approach, although their idea can be used with any proximity-based clustering algorithm (any algorithm based on a proximity matrix). The original proximity matrix is modified in order to satisfy the available constraints and their implications, i.e. the constraint propagation, which are achieved when clusters are merged.

Partitional clustering in general, and K-means in particular, have received much attention by including available constraints. One of the first example is COP-K-means [256]. The basic idea of this algorithm is based on to modify the traditional K-means algorithm taking into account that constraints are never violated when updating cluster assignments. PC-K-means [17] is similar to COP-K-Means, but this algorithm does not enforce to satisfy all the constraints. Also related to K-means, Basu et al. [16] proposed not to use the available constraints to guide the clustering process, but to generate initial seed clusters which give prior information about the final clustering.

Examples of algorithms based on learning a new metric are M-K-means [18], based on traditional K-means, and [44], based on learning a new metric for fuzzy C-means. Xing et al. [268] also presented a similar algorithm, which learns new metrics but does not require that the constraints must be satisfied.

One of the most known hybrid algorithms based on K-means is metric pairwise constrained K-means (MPCKM) [28], which greedily optimizes an objective function ($\mathcal{J}_{MPCKM}$), based on both pairwise constraint satisfaction and metric learning depending on instance relations. This algorithm is also presented and compared to some of the previously presented algorithms in [18]. MPCKM is used throughout this thesis, so some details are provided. $\mathcal{J}_{MPCKM}$ must

be minimized and is defined as

$$\begin{aligned}\mathcal{J}_{MPCKM} = & \sum_{\mathbf{x}^{(i)} \in \mathcal{X}} (\|\mathbf{x}^{(i)} - \boldsymbol{\mu}_{l_i}\|^2_{\mathbf{A}_{l_i}} - \log(\det(\mathbf{A}_{l_i}))) \\ & + \sum_{(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \in \mathcal{M}} w_{ij}^{\mathcal{M}} f_{\mathcal{M}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \mathbb{1}[l_i \neq l_j] \\ & + \sum_{(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \in \mathcal{C}} w_{ij}^{\mathcal{C}} f_{\mathcal{C}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \mathbb{1}[l_i = l_j].\end{aligned} \tag{2.27}$$

The specific notation is as follows. Let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ be two instances of a data set $\mathcal{X}$, and $\boldsymbol{\mu}_{l_i}$ be the centroid of the cluster l_i assigned to $\mathbf{x}^{(i)}$. Then, $\|\mathbf{x}^{(i)} - \boldsymbol{\mu}_{l_i}\|^2_{\mathbf{A}_{l_i}}$ is the parametrized Euclidean distance using a weight matrix $\mathbf{A}_{l_i}$ from the instance to the specific centroid. The second term of the first addend is due to normalization with covariance matrix $\mathbf{A}_{l_i}^{-1}$. Regarding the constraints, $\mathcal{M}$ is a set of must-link constraints and $\mathcal{C}$ is a set of cannot-link constraints. For each pair of instances, $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, there are two penalty costs for violating these two kinds of constraints, $w_{ij}^{\mathcal{M}}$ and $w_{ij}^{\mathcal{C}}$, respectively. The penalty also depends on the distance between the instances, so two functions $f_{\mathcal{M}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ and $f_{\mathcal{C}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ are used for this purpose. Finally, $\mathbb{1}$ is an indicator function, where $\mathbb{1}[true] = 1$ and $\mathbb{1}[false] = 0$.

Other examples of hybrid algorithms are hidden Markov random field (HMRF)-K-means [19] and the adaptive semi-supervised clustering Kernel method based on metric learning (SCKMM) [273]. The former incorporated available constraints along with an underlying distortion measure into K-means by using HMRFs. And the latter used K-means for clustering incorporating constraints and using an objective function which was also obtained from constraints estimating the parameter of a Gaussian kernel.

Cohn et al. [55] obtained incorporated constraints in order to satisfy user feedback. This feedback is iterative and satisfied in each new iteration. These authors incorporated this knowledge into a model-based clustering approach, by augmenting the standard Kullback-Leibler (KL) divergence [161] with a weighting function, but reiterated that the approach is, in theory, applicable to almost any clustering algorithm.

The idea of using partially labeled data sets together with EM was introduced in [192]. Shental et al. [236] also introduced constraints into the EM algorithm for finite mixture models using a closed-form EM procedure with only must-link constraints, and a generalized EM procedure with Markov nets to estimate the model including also cannot-link constraints. This was also applied to a leukemia data set in [8]. The inclusion of clustering constraints into model-based clustering can also be found in [162], where Law et al. modified the EM algorithm by treating the constraints as random variables. In the same year, Lu and Leen [177] expressed the clustering constraints in the prior distribution over assignments of instances to clusters in a Gaussian mixture model. They penalized the cluster assignments depending on this prior and according to the degree of constraint violations.

Other algorithms based on different clustering approaches are [35] and [69], where the objective function is somehow regularized depending on the label information to obtain the

final clustering based on GAs and the fuzzy C-Means, respectively. Again with the restriction of satisfying all the constraints, Xia [267] presented a global optimization method to solve the semi-supervised clustering problem.

An example of an algorithm based on density-based clustering is C-DBSCAN [230], which extended DBSCAN by first partitioning the data into dense spaces. From them, local clusters are identified and merged later according to the must-link constraints. Finally, cannot-link constraints are satisfied to obtain the final clusters.

Baghshah and Shouraki presented two papers [12, 13] based on metric learning for semi-supervised clustering using both available constraints and the geometrical structure of data to improve the performance of semi-supervised clustering algorithms.

Liu et al. [174] presented a completely different approach, in a framework called Boost-Cluster. The aim of this framework is to improve the accuracy of the clustering solution by exploiting the constraints, independently of the used clustering algorithm. This is an iterative process in which new data are generated in each iteration, depending on the clustering results at previous iterations and on the available clustering constraints.

2.5 Dimensionality reduction

Advances in data acquisition and storage have not only led to large amounts of data, but also to data with large number of features. This fact, that could be worthy on paper, is not always desirable at learning because data with a large number of features can become unhandled data by pattern recognition algorithms. Besides, even though an algorithm might be able to deal with large amounts of features, some of them can be either irrelevant or redundant for the task. An attribute is irrelevant for the pattern recognition task when it does not contribute to improve the final result or even degrades it, according to the used performance measure. On the other hand, an attribute is redundant when it contributes to the task with the same information than another attribute. These facts are very related to the well-known *curse of dimensionality*, cited by Bellman [22] as "[The] curse of dimensionality [...is] a malediction that has plagued the scientists from earliest days.". The summarized explanation of this concept, related to any optimization problem, is that the difficult of an optimization problem increases exponentially with an increasing number of features. This fact becomes even worse when the number of instances is not increased together with the number of features.

Feature subset selection

For the above reasons, it is often desirable to select a subset of features before, or sometimes during, performing the learning task. This selecting task is called FSS and has been widely studied in data mining [10, 171]. FSS can be seen as a process in which several subsets are searched, and the best of them is then selected based on some evaluation criterion. Therefore, two different steps must be performed: search and evaluation.

Regarding the search, it can be seen as a problem in a space search of 2^F different possibilities, with F the number of available features. Some optimal searches, like depth-first

and breadth-first (which are typical search algorithms in graph theory for instance), are based on exhaustively look for all possible combinations, finding the best possible subset. These techniques cannot be applied when the number of features F becomes relatively high since the space search size becomes too large. Heuristic techniques are then applied trying to find solutions close to the optimal, and, in some situations, even the optimal itself. Heuristic approaches can be divided into:

- Deterministic. This kind of algorithms obtains the same solution for each run and configuration. Most of these algorithms are "greedy", i.e., from a starting point, the search continues until the optimization function does not improve once. The two most common algorithms are forward selection and backward elimination [150]. The former starts with no features and one feature is added in each step, whenever the objective function improves. On the other hand, backward elimination starts with the whole set of available features and one of them is deleted in each step, until the convergence criterion is reached, i.e., the objective function does not improve when any instance is deleted.

- Stochastic. These algorithms might vary their execution using some kind of randomness. Therefore, solutions vary for different runs. GAs are an example of these kinds of algorithms [245, 270], since their search step evolves good feature subsets by using random perturbations of a current list of candidate subsets.

Besides the search step, it is necessary to define the objective or evaluation function that must be optimized to select the features subset in the evaluation step. Regarding this, FSS has been faced from two different approaches [171]:

- *Filter*. This approach evaluates data properties before the learning task. Therefore, the evaluation relies on data characteristics without using any learning algorithm. Filter algorithms rank features according to a measure, e.g., correlation between features [118] or RELIEF [149].

- *Wrapper* [153]. This approach identifies feature relevance based on the evaluation performance of a learning algorithm. Therefore, the search objective function is the same than evaluating the applied learning algorithm. Using this approach can lead to bias depending on the learning algorithm and it is more computationally demanding.

Some supervised classification algorithms, such as some decision tree-based, also select some features as relevant during the classification process. This can be seen as another approach, called *embedded*. One of the most known examples of algorithm using this approach is the classification tree algorithm C4.5, previously presented. Liu and Yu [173] also referred to another approach, called *hybrid*, in which filter and wrapper algorithms are used at different stages.

Some of the last FSS approaches, together with potential future work, can be found in [172]. Therefore, FSS continues as an active research line in data mining due to the importance of this process in real domains such as genomics [133], text mining [89], or intrusion detection [164].

Feature subset extraction

If a feature subset is selected from the original features in FSS, FSE is a process in which new features (different from the original ones) are created. Therefore, new features are extracted from the available feature set, getting a lower dimensional space. principal component analysis (PCA) [141], one of the most widely used FSE algorithms, obtains new uncorrelated variables named principal components (PCs), which preserve as much of the original information as possible. These PCs are sought from the original features and maximize the data variance captured. It is a mathematical procedure and can be calculated from the eigenvalue decomposition of the data covariance matrix.

FSE is out of the scope of this thesis, but other examples of general techniques are wavelet transforms [59], which provides very general techniques that can be applied to many tasks in signal processing, allowing the possibility to manipulate data to obtain compressed parameters, and partial least squares (PLS) [263], which is a class of methods for modeling relations between sets of observations by means of latent variables.

2.5.1 Subspace clustering

Focusing on clustering, the FSS process becomes in a more difficult problem when each cluster may have a different subset of relevant features than other clusters. This problem is called *subspace clustering* and an illustrative motivation to it, using figures extracted from [205], follows.

Figure 2.8 shows a 3-dimensional representation of a data set separated into four clusters. The main characteristic of these data is that instances from each cluster were generated using a feature subset. Clusters represented in red and green colours were generated using features a and b, whereas clusters in blue and purple colours were generated using features b and c. When clusters are characterized using different features and a traditional clustering algorithm is used to partition the data, the common situation is that the obtained partition achieves a poor result. This problem becomes even worse when the number of features is increased.

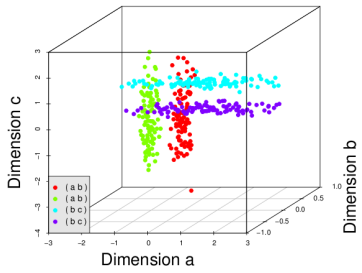
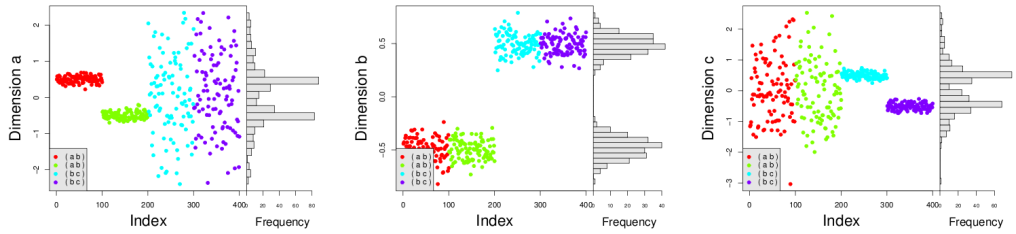


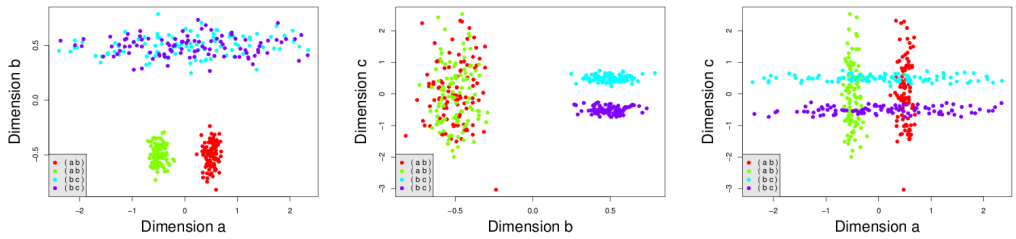
Figure 2.8: Data set with four clusters represented in three dimensions. Each cluster, generated using only a subset of the available features, is represented by a different colour. This figure was taken from [205].

An FSS technique applied to this data set does not fix the problem. This can be seen in Figures 2.9 and 2.10, where data are represented in one and two dimensions, respectively,



(a) Dimension a (b) Dimension b (c) Dimension c

Figure 2.9: Data set plotted in only one dimension, including a histogram. Different clusters are again represented using different colours. This figure was taken from [205].



(a) Dims a & b (b) Dims b & c (c) Dims a & c

Figure 2.10: Data set plotted in two dimensions, covering all possible combinations that could be selected. This figure was taken from [205].

covering all possibilities that can be obtained using an FSS technique. Using neither only one feature nor any combination of two features would be enough to discover the four data structures as can be seen in both figures.

Nevertheless, clusters are completely visible and separated in pairs, as can be seen in Figure 2.10. In 2.10 (a), where features a and b are used to plot the data, clusters green and red are separated. In 2.10 (b), where features b and c are used to plot the data, clusters blue and purple are completely distinguished. This matches with how each cluster was generated. Therefore, searching in different feature subsets (subspaces) leads to the correct clustering solution.

In sum, and as detailed in this chapter, a clustering task can be performed: (i) without any dimensionality reduction process, i.e., using the original and available data, (ii) obtaining a single feature subset either by selecting features, i.e., FSS, or by extracting new ones, i.e., FSE, and (iii) identifying several subspaces to characterize each cluster. These possibilities are shown in Figure 2.11.

As commented above, the general problem of finding clusters characterized by different dimensions is roughly called subspace clustering. Besides [205], there are some surveys on subspace clustering [197, 200, 208, 237], where different approaches and algorithms are discussed and compared. Although these surveys are a good source of information about different approaches, Kriegel et al. [160] presented another survey focused on giving a more systematic approach to the problem. These authors, like others, directly related this kind of problem to

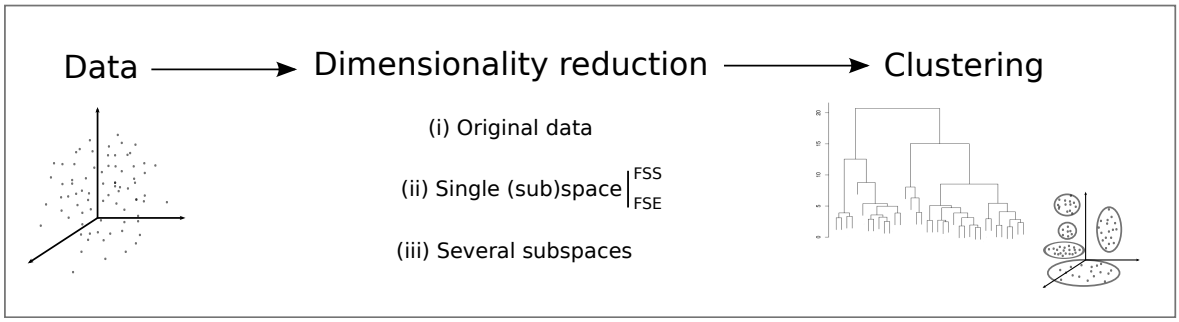


Figure 2.11: Possibilities regarding the data to perform a clustering task.

high-dimensional data. It is obvious that high-dimensional data are clear targets for finding clusters using different feature subsets due to the number of available features, but this thesis does not limit this kind of problems only to high-dimensional data, since subspaces may exist in data with fewer features. Besides, and citing to [120]: "Note that high-dimensional can be as few as $p = 10$ variables or as many as $p = 1000$ variables or beyond - it depends on the complexity of the models concerned and on the size of the available data".

The categorization of subspace clustering approaches depends on the paper or survey consulted, since different authors categorized these approaches using different criteria. One of the most complete categorizations can be found in [160], and therefore, the categorization that follows is based on that survey, However, names and concepts that other authors also used are included when appropriate.

A category, different to the used in this work, is called by [160] as "finding clusters based on patterns in the data matrix" (also called "biclustering" and "coclustering"). This kind of algorithms cluster instances and features interchangeably. For a related survey, see [180]. The next necessary distinction to categorize different subspace clustering approaches is whether clusters can be found in arbitrarily oriented subspaces or only in axis-parallel subspaces. The former case, called "correlation clustering", is a more general case, but it is necessary to take into account that there is an infinite number of arbitrarily oriented subspaces, which leads to computationally infeasible problems unless heuristics are used. It can be enough, depending on the application, to assume only the existence of axis-parallel subspaces. An example of axis-parallel subspaces can be seen in Figure 2.10. Many works are focused on searching subspaces under this assumption, and different approaches can be found in this sense. The difference between the orientation of the subspaces is not taken into account by some authors, like in [205] for instance.

There are two possible categorizations of axis-parallel subspaces approaches. The first one, although only used by some authors, is based on a problem-oriented categorization:

- Projected clustering algorithms. Each instance belongs to only one cluster, which is characterized using a subspace. Therefore, there is not overlapping in clusters (hard clustering). Found subspaces are those that best cluster data according to some definition of what a good cluster is. Related to it, there is another tendency, called "soft projected clustering", in which subspaces are not found by selecting (retaining or dis-

carding) features, but by weighting them. A feature with a high weight is considered as more important to define a subspace than another lower weighted feature. This tendency is merely called "soft subspace clustering" by some authors [71, 138], and is similar to "weighting clustering" but extending it to every single cluster. Thus, it is a multiple feature weighting clustering [202]. Deng et al. [71] divided this approach further into "fuzzy weighting subspace clustering" [45, 99, 98, 139] and "entropy weighting subspace clustering" [74, 138], depending on whether the algorithms are based on either assigning a fuzzy weight to each feature and cluster, or those weights are controlled depending on the entropy, respectively.

- Subspace clustering algorithms. The aim is to identify all subspaces where clusters can be found. Therefore, instances can be grouped in many different clusters, each characterized by a different subspace. Therefore, the overlapping is allowed by algorithms from this category.

- Hybrid algorithms. As its name suggests, this approach is a hybrid between the two previously presented. Found clusters may overlap, but the number of found subspaces is not the full set of possible subspaces, but only a smaller subset according to some criteria.

This categorization[1], presented in [160], is also followed in [237] for instance, but it does not appear in previous surveys, such as [205]. This categorization can be confusing because the concept "subspace clustering" is not always used in the same manner. See for instance [277], where Yiu and Mamoulis used projected clustering and subspace clustering as the same concept.

The second categorization (algorithmic-oriented) presented in [160] is the same as the one used in [205]. Therefore, the agreement with this categorization is complete:

- Bottom-up approach. It is based on finding dense regions of data using low dimensional spaces. These regions are then combined until a cluster can be formed. This approach is based on the downward closure property of density, which states that if there are dense units in a number of dimensions, F, there are dense units in all $(F-1)$ dimensional projections. One of the problems of this approach is the amount of necessary input parameters, e.g., the density threshold, and the difficulty of properly tuning them.

- Top-down approach. This approach tries to find a clustering result using all available dimensions. After this, subspaces of each cluster are searched and new clusters must be found. Therefore, a circular dependency begins. To escape from this, most top-down algorithms are based on the *locality assumption* [160]. This assumes that the subspace of each cluster can be learned from the local neighborhood of cluster members.

[1] According to this categorization, the title of this thesis should be "Semi-supervised projected clustering and application to neuroscience". However, due to this categorization depends on the authors, the title maintains the general concept "subspace clustering" as a more identifiable concept

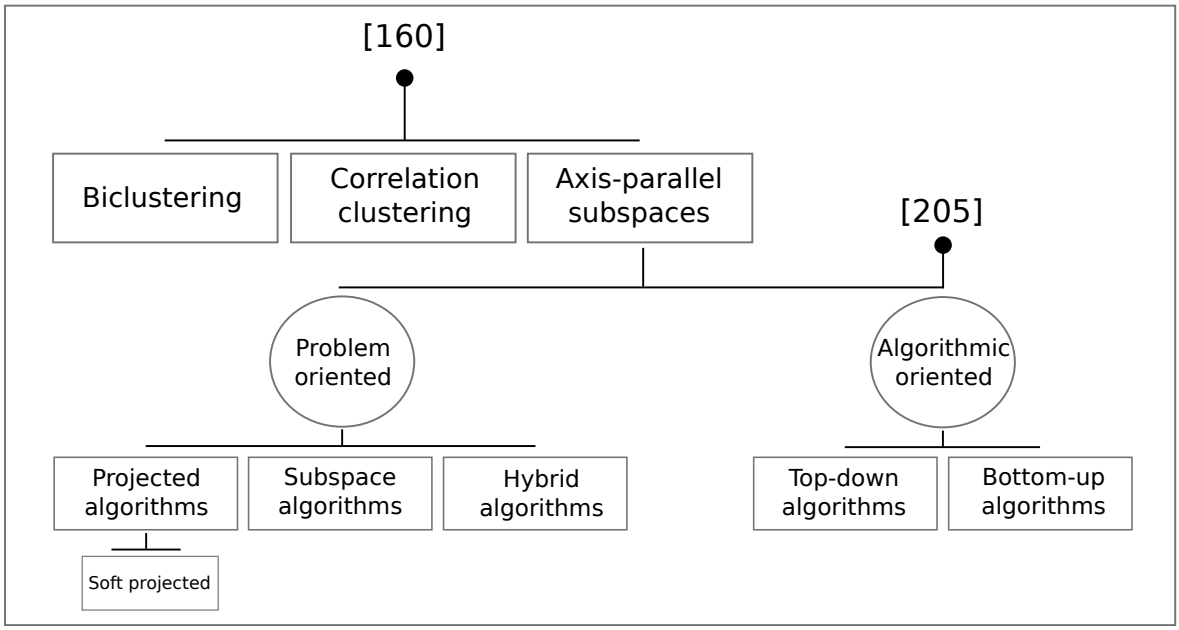


Figure 2.12: General schema of approaches that search for several subspaces for clustering data according to [160] and [205].

The general schema of approaches that search for several subspaces for clustering data is illustrated in Figure 2.12. The problem-oriented and the algorithmic-oriented categorizations are completely different. However, they have some relations when algorithms are seen in detail. All subspace clustering algorithms are implemented using a bottom-up approach, whereas most projected clustering are top-down algorithms. These direct relations between both categorization types are taken into account in [119], where algorithms were divided into "dimension-reduction projected clustering" and "dimension-growth subspace clustering". Nevertheless, this generalization should be made wisely, because some projected clustering algorithms are implemented using a bottom-up approach. On the other hand, hybrid algorithms do not appear in those relations and some of these algorithms are implemented using a bottom-up approach and some others using a top-down approach.

According to these categorizations, a presentation of some of the most known algorithms that find clusters in several subspaces follows. Besides, a summary of all the algorithms that search for axis-parallel subspaces, depending on the category, can be found in Table 2.3.

	Problem-oriented		
Algorithmic-oriented	Projected (Soft projected*)	Subspace	Hybrid
Top-down	[2, 265, 33] [93, 75]*		[214, 274]
Bottom-up		[4, 48, 53, 103, 143, 169, 170]	[1, 159, 235]

Table 2.3: Summary of some of the most known algorithms that find clusters in several subspaces regarding the previously presented categorizations.

Projected clustering (PROCLUS) [2] was the first projected and top-down algorithm. This

algorithm is close to K-means (see Section 2.3.1) and uses an iterative process that starts determining some cluster centers using a greedy algorithm. Cluster subspaces are searched then by minimizing the distances standard deviation of the instances in the neighborhood of each cluster center to the corresponding cluster center in each dimension. Finally, instances memberships are estimated depending on these cluster centers and subspaces. An extension of PROCLUS is an algorithm called fast and intelligent subspace clustering algorithm using dimension voting (FINDIT) [265], which employs additional heuristics to enhance efficiency and accuracy of PROCLUS.

Subspace preference weighted density connected clustering (PreDeCon) [33] is also categorized as a projected algorithm and is based on DBSCAN. This algorithm builds a subspace preference vector, based on the variance in each feature, and uses a weighted distance measure according to the vector. Although PreDeCon was presented as an efficient algorithm, robust to noise, and linear in the number of features, there are four input parameters, related to density and preference vector, that must be correctly tuned to obtain good performance.

Clustering on subsets of attributes (COSA) [93] is another top-down algorithm. This algorithm is different from the previously presented because it does not cluster the data, but obtains a similarity matrix, examining the K-nn of each instance. The similarity matrix can be used to cluster data with another algorithm later. This matrix contains information between instances and dimensions using weights. Hence, subspaces are found for each instance instead of for each cluster, and when the cluster task is performed, cluster subspaces must be estimated depending on the instances of each cluster. Although COSA is categorized by Parsons et al. [205] as a top-down algorithm, these authors do not take into account soft subspace clustering algorithm, and COSA is categorized as a soft subspace clustering algorithm by [138] and as a soft projected clustering algorithm by [160].

Another example of soft projected clustering algorithm is locally adaptive clustering (LAC) [75]. The search for subspaces is based on discovering clusters using different combinations of features via local weightings. Weights are configured depending on the correlation of instances in each cluster. LAC is based on K-means and EM (see the relation between K-means and model-based clustering in Section 2.3.1). There is an initialization and, after that, an iterative process firstly assigning instances to clusters and recalculating clusters. The difference between LAC and other similar algorithms is related to the weight formulas that result from different objective functions to be minimized in the clustering process. An important and related concept is *feature saliency*, introduced in [163], that is the probability that a feature is relevant to a cluster. Therefore it is a close concept to soft subspace clustering if it is extended to all clusters separately. Law et al. [163] introduced this probability into the EM algorithm for model-based clustering, getting that irrelevant features got a saliency of 0. The extension of [163] to subspaces was presented by Li et al. [166], using also minimum message length (MML) criterion to estimate the final number of clusters. Besides, the same authors presented a variant in [167], where clusters and individual feature subsets for each cluster were found simultaneously through parameter estimation using variational Bayesian learning. Previous to this work, and also based on variational approximation, Ghahramani

and Beal [102] presented an algorithm that infers the model structure of a mixture, also determining the number of components and the local dimensionality of each cluster.

A different approach, presented by Graham and Miller [110], was based on shared distributions obtaining more parsimonious models. This is close to projected clustering because each cluster has its own specific feature subset distribution, but all of them share a common distribution for some features when it is possible. This method obtains more flexible models trying to overcome the "structural failure", i.e., the problem of estimating the number of clusters by varying the complexity of a model in terms of degrees of freedom.

Hoff [127] developed an approach, based on a multivariate Dirichlet process mixture model, that estimated the number of clusters, the clustering partition, and the specific parameters for each cluster, in a unified way. This proposal was thought for binary features, specifically for the analysis of genomic abnormality data. Hoff reported a similar research with fewer limitations in [128]. Other works, also for non-Gaussian data, based on generalized Dirichlet process are [36, 37]. Again for binary data, Patrikainen and Mannila [207] presented SUBBERCLUST, a subspace mixture model algorithm for Bernoulli distribution. A recent proposal can be found in [38], where a new generalized Dirichlet mixture model is used to address the problem of clustering multidimensional data sets on different subspaces.

All subspace clustering algorithms are also bottom-up algorithms. The first created algorithm was clustering in quest (CLIQUE) [4], which was not only the first bottom-up algorithm but also one of the first algorithms that found clusters in different subspaces. CLIQUE is a density and grid-based clustering algorithm that iteratively finds cluster combining adjacent dense grid units, which are defined in different subspaces depending on the coverage, i.e., the fraction of data covered by the units.

There are several variants of CLIQUE, all of them categorized as subspace and bottom-up algorithms. The first variant is called entropy-based clustering (ENCLUS) [53]. The main difference between CLIQUE and ENCLUS is that the latter is based on entropy instead of density. ENCLUS assumes that a subspace with clusters has lower entropy than a subspace without clusters. Both CLIQUE and ENCLUS use a static sized grid to split each dimension in bins, therefore this is another input parameter that must be fixed before running these algorithms. Trying to handle this fixed size, other algorithms introduced different techniques to automatically determine, in each case, the number of bins for each dimension. merging of adaptive finite intervals (MAFIA) [103] splits each dimension depending on the data distribution. Cell-based clustering [48] and CLTree [169] are other two examples of algorithms with an adaptive grid size, based on examination of min-max values of each dimension, and on a decision tree, respectively. Another variant of CLIQUE is called nCluster [170]. This algorithm does not use a grid based approach to partition the data into non-overlapping rectangle cells, but uses a more flexible method to partition the dimensions to preserve meaningful and significant clusters.

Density-connected subspace clustering (SUBCLU) [143] is another subspace and bottom-up algorithm based on DBSCAN that satisfies the downward closure property. This algorithm is able to detect arbitrarily shaped and positioned clusters in subspaces, therefore it is more

accurate than grid-algorithms but has a high computational cost. This algorithm can be summarized as if a DBSCAN run would be applied for each identified subspace.

There are several algorithms categorized as hybrid algorithms by [160]. Density-based optimal projective clustering (DOC) [214], which is density-based, as some other algorithms commented above, is one of them. DOC process is very flexible, since several decisions can be made depending on the requirements. The process finds, for each run of the algorithm, an approximation of an optimal cluster over the current instances. Therefore, if instances that were clustered in the first iterations are excluded for the following iterations, DOC will not find overlapped clusters. DOC could be considered a pure projected clustering algorithm in this case. On the other hand, if all instances are always maintained as input, overlapped clusters may be found. This possible variation categorizes DOC as a hybrid algorithm.

Hierarchical approach with automatic relevant dimension selection for projected clustering (HARP) [274] and detecting subspace cluster hierarchies (DiSH) [1] are hybrid algorithms based on hierarchical clustering. The former, which is a top-down algorithm, does not find a hierarchy of subspace clusters but uses different distance functions to iteratively merge clusters minimizing the number of relevant attributes for each cluster. On the other hand, DiSH is a bottom-up algorithm and based on the key concept of "subspace distance", that assigns small values if two instances are in a common low-dimensional subspace cluster and high values if they are in a common high-dimensional subspace cluster or are not in a subspace cluster at all. Subspace clusters with small subspace distances are embedded within clusters with higher subspace distances. According to this, DiSH uncovers hierarchies of nested subspace clusters.

Another example of hybrid bottom-up algorithm is support and Chernoff-Hoeffding bound-based interesting subspace miner (SCHISM) [235], which is based on mining interesting subspaces, i.e., this algorithm searches for subspaces that can be used for finding clusters subsequently. Filter refinement subspace clustering (FIRES) [159] is a framework that can be used with any kind of cluster approach. Firstly, clusters are found using only one dimension and are then merged to find maximal-dimensional subspace cluster approximations. There is a last step to refine these cluster approximations. This framework is also considered bottom-up.

Projected clustering via cluster cores (P3C) [196] is another hybrid algorithm that also starts by finding clusters using one dimension. Then, "cluster cores" are identified by combining the previously found regions. Finally, these "cluster cores" are refined for finding the final clusters. P3C is also effective in detecting clusters with varying orientation in their relevant subspaces. This type of algorithms is illustrated in Figure 2.12 as "arbitrarily". Another typical example of this kind of algorithms is oriented projected cluster generation (ORCLUS) [3], which is an extension of PROCLUS. Many of these algorithms are based on FSE using PCA. Related to this, although out of the scope of this thesis, Tipping and Bishop [247] reduced data dimensionality, based on PCA, obtaining a mixture model for probabilistic principal component analysers using the EM algorithm.

Once some of the most known subspace and projected clustering algorithms are presented, algorithms using some kind of knowledge to enhance the search for subspaces and the

clustering process are detailed.

Subspace clustering and constraints

There are some works dealing with subspace clustering and using some kind of available information, like constraints, to improve the final result. In 2005, Yip et al. presented semi-supervised projected clustering (SSPC) [275], based on a partitional method similar to K-medoids algorithms to find projected clusters. These authors used the term projected clusters to identify clusters characterized by only small subsets of all available features. Besides, this algorithm is able to utilize domain knowledge (labeled instances) to improve the clustering accuracy by determining the initial seeds of the search space. According to this work, this is the first semi-supervised projected clustering algorithm. A similar work from some of the same authors was extended and applied to microarray data in [276].

Cheng et al. [54] presented an extension of LAC, called constrained locally weighted clustering (CLWC), incorporating pairwise constraints (must-link and cannot-link constraints) into the local weighting scheme, according to the concept of "chunklet", i.e., subset of instances that are known to belong to the same although unknown class [15].

Fromont et al. [95] proposed to extend bottom-up subspace clustering algorithms, like CLIQUE, by integrating the available known information. These authors presented an algorithm, called SC-MINER, for subspace clustering. The algorithm is based on extending the classical steps of bottom-up subspace algorithms (pre-process, mining, and post-process) by including constraints. The pairwise constraints are integrated into the mining step. During the same year, Ahmed and Khan [5] presented semi-supervised impurity based subspace clustering (SISC), used in conjunction with K-nn, and applied to text classification. This algorithm, based on the EM algorithm, is able to identify subspaces including the *Chi Square Statistic* in the objective function. Besides, another component called *impurity* is also used to modify the *dispersion* measure [186] for each cluster. Finally, K-nn is applied using the subspace cluster centroids as neighbors to finally assign labels.

Zhang et al. [280] pointed out that feature correlation and distance divergence are important considerations for subspace clustering. Feature correlation can be used to reduce the search space, whereas distance divergence can be used to distinguish distances on different dimensions and to find clusters accurately. According with these two aspects, they presented a top-down subspace clustering algorithm, called constraint based dimension correlation and distance divergence (CDCDD), based on the algorithm previously presented FINDIT, that uses the aid of available constraints. The same authors presented another algorithm called S^3C [279] in a similar line than the previous one, and also exploiting available constraints. This algorithm first identifies subspaces by finding consistencies according to the available constraints. Once the subspaces are found, instances are assigned to clusters based again on constraints and found subspaces.

This is the up to date state-of-the-art regarding subspace clustering algorithms using available constraints. Based on these works, using available information might improve clustering solutions either helping to firstly identify subspaces before the clustering assignments

(like in the proposal presented in Chapter 4), or during the clustering process directly (like in the proposal presented in Chapter 5).

Part III

PROPOSALS

Chapter 3

Clustering validation indices

3.1 Introduction

As commented in Section 2.3.2, clustering validation is a very difficult task due to the lack of ground truth in clustering. There are many CQIs that can be used to validate a clustering solution [193], but research studies about these indices are often based on utopian data conditions, with low dimensionality and clear separation and cohesion of clusters. Besides, in this type of works, only one clustering approach is often used, but different indices may have different behaviours depending on the way the solutions were obtained. For these reasons, a study of some of the most known CQIs using different data conditions and clustering approaches is presented throughout this chapter. A shorter version has been published in [114].

This study focuses on using each CQI as a stopping rule for finding the real number of clusters for each particular solution, which is an important and common step in clustering. This type of validation is known as relative criterion and was commented in Section 2.3.2. Besides, close-to-real domains for the purpose of evaluating, indices are used dealing with different data characteristics and clustering algorithms. Thus, some of the most used CQIs are compared here in different scenarios to output some behaviour patterns that can help decision making on what index should be used depending on the problem at hand and how it is solved. The scenarios are created using different databases that aim to simulate real cases, having different percentages of outliers or noisy dimensions. Data are partitioned using three different clustering approaches and one random algorithm to check whether each index behaviour is different in each case. Finally, and following [252], an external CQI is used as support for the validation.

Chapter outline

The next section presents the algorithms and indices used during this chapter, although details about both algorithms and indices can be found in Chapter 2. Then, Section 3.3 covers all the experimental studies, presenting the used data, the validation process, and the

obtained results. Finally, the chapter ends with a summary and some discussion in Section 3.4.

3.2 Algorithms and indices

Three clustering algorithms from different approaches (see Section 2.3) are used to partition each data set: K-means, hierarchical clustering using Ward's method, and model-based clustering using Gaussian mixtures and the EM algorithm. All these algorithms are prime examples of different approaches and, therefore, are well-known in clustering literature. Apart from using these algorithms, data are also randomly partitioned as if it were another algorithm. For each data set, this random strategy is executed for 50000 iterations in an attempt to achieve statistical significance for each case.

The study compares five internal CQIs, using one external CQI to check whether clustering algorithms are able to identify the correct cluster structure for each data set. The five internal CQIs that have been introduced in the comparison are: Silhouette, Calinski, C-index, DB, and Gamma. These indices were previously detailed in see Section 2.3.2. Silhouette is the only index that was not compared in [193], but it is widely used in different fields, like, for example, genomics [176] or neuroscience [145]. On the other hand, ARI is used as external CQI. This index was created as an improvement on the Rand index. Both indices belong to pair counting based measures, as explained in Section 2.3.2.

The indices compared in this work were designed to be used together with hard clustering algorithms. Thus, the soft clustering solutions that are obtained with model-based clustering are adapted by assigning each instance to the cluster whose probability of membership is maximum.

As a note about indices, remark that there are many other CQIs that are not considered in this work, some examples are Dunn index [79], Je(2)/Je(1) [76], or Beale index [21]. It is not possible to cover all the available indices, but the selected ones are either well-ranked instances according to [193] or widely used indices in clustering validation literature.

3.3 Experimental results

3.3.1 Data

The differences in the data used in this comparison are related to impurities, such as noisy dimensions and outliers. All these details are presented in the following and summarized in Table 3.1.

The data sets were generated using the original cluster data generator software described in [193]. All data sets are detailed in the following, where they are divided into three groups. The first group (*clear*) is composed of data sets with strong and distinct clusters. The second group (*out5* and *out10*) has data sets generated with 5% and 10% outliers, respectively. Outliers are instances that do not belong to any predefined cluster. Finally, the last group

(*noi1* and *noi2*) has data sets with 1 or 2 added random noisy dimensions, respectively. A noisy dimension is a feature that does not contribute to the correct clusters separation.

The number of data sets in each group depends on the number of dimensions, clusters, and density levels. The density levels were designed to change the cluster sizes and the instance distributions. At the first density level each cluster has the same number of instances; at the second level, one cluster always contains 10% of instances; and at the third level, one cluster contains 60% of instances. At last, 228 data sets are used in the comparison:

Data	F	K	DL	N
clear	4,6,8	2,3,4,5	1,2,3	50
out5	4,6,8,10	2,3,4,5	1,2,3	105
out10	4,6,8,10	2,3,4,5	1,2,3	110
noi1	5,7,9,11	2,3,4,5	1,2,3	100
noi2	6,8,10,12	2,3,4,5	1,2,3	100

Table 3.1: Summary of databases, where F is the number of features, K is the number of clusters, DL are the different density levels, and N is the number of instances of each data set. For more details about each data set, see the text.

1. There are 36 data sets in the first group, resulting from combining four different number of clusters (from 2 to 5 clusters), each with different number of features (4, 6, and 8), and three different density levels. The number of data instances in each data set from this first group is 50.

2. The second group is divided into data sets with 5% and 10% outliers. There are 48 data sets in each subgroup since a new dimensionality (10 features) is added on top of all the combinations explained for the first group. Besides, 105 and 110 instances are used in each data set depending on if there are 5% or 10% outliers, respectively.

3. The last group has data sets with 100 instances each, but one noisy dimension is added to *noi1* and two noisy dimensions are added to *noi2* for each original number of features. The total number of data sets is again 48 for each subgroup.

3.3.2 Evaluation process

The methodology consists of creating clustering partitions, using K-means, hierarchical clustering, and model-based clustering algorithms, and the random grouping algorithm for all the possible number of clusters (from 2 to 5) for each data set and using the CQIs to evaluate each built partition. Thus, by using four different algorithms and four number of clusters combinations for each data set, $228 * 4 * 4 = 3648$ partitions are evaluated with five internal and one external CQIs. The external CQI, ARI, is used as external validation to assess the quality of each built partition against the real partition, which is known beforehand.

For each index evaluation, the best number of clusters is the maximum or the minimum index value depending on each CQI. This choice will be correct if the chosen number of clusters matches the real number of clusters known beforehand. Otherwise, the choice will be classed as wrong irrespective of the distance to the real number of clusters. Besides, by evaluating the external CQI, it is possible to find out whether the clustering algorithms are

able to find the real cluster structures in data regardless of correct or incorrect internal CQIs number of clusters choices.

3.3.3 Results

Table 3.6 shows the number of correct decisions on the number of clusters output by each CQI and each clustering algorithm for the 228 data sets. The number of correct decisions output by ARI is very high. This means that the clustering algorithms are able to find the correct cluster structure in many situations, especially when they have to find 2 or 3 clusters. Regarding internal indices, Calinski achieves the best results with around 70% correctly decisions. It is followed by Silhouette, DB, and Gamma, which all achieve very similar results. C-index is the clear loser in this first comparison. It is interesting to note however that this index is at least as competitive as Silhouette, DB, and Gamma at finding 5 clusters.

		K					
Index	Algorithm	2	3	4	5	Total	%
Silhouette	K-means	31	31	29	33	124	54.386
	Hierarchical	38	39	35	32	144	63.158
	EM based	39	33	31	33	136	59.649
Calinski	K-means	50	40	28	42	160	70.175
	Hierarchical	51	39	29	40	159	69.737
	EM based	50	37	33	36	156	68.421
C-index	K-means	22	14	13	31	80	35.088
	Hierarchical	9	8	7	35	59	25.877
	EM based	6	4	7	39	56	24.561
DB	K-means	27	31	28	31	117	51.316
	Hierarchical	30	35	31	28	124	54.386
	EM based	41	32	25	21	119	52.193
Gamma	K-means	21	32	29	36	118	51.754
	Hierarchical	23	36	30	36	125	54.825
	EM based	22	30	27	41	120	52.632
ARI	K-means	55	55	52	46	208	91.228
	Hierarchical	57	55	52	47	211	92.544
	EM based	55	55	54	49	213	93.421

Table 3.2: Number and percentage of correct decisions for each CQI, clustering algorithm, and number of clusters K (from 2 to 5) in 228 data sets: 36 with clear cluster structures, 96 with outliers, and 96 with noisy dimensions.

Regarding the clustering algorithms, K-means behaves better when used with Calinski and C-index, whereas hierarchical clustering outperforms the other algorithms when used with Silhouette, DB, and Gamma. Interestingly, EM is the best algorithm only when used with ARI, which is the most precise index.

Indices behaviours in data sets with outliers

Results for the 96 data sets with 5% and 10% outliers are shown in Table 3.3. As expected, these results are different from the results shown in Table 3.6. In this case, the biggest differences depend on the used clustering algorithm. For example, there is a 20% difference in the number of correct decisions obtained by Silhouette depending on using either K-means or hierarchical clustering. This also applies to C-index, because EM returns around 26% and K-means around 51% correct decisions. The internal CQI with the highest percentage of correct decisions regardless the clustering algorithm is again Calinski. Taking into account all three clustering algorithms, Gamma mean results are a worse than C-index mean results. C-index achieves better results than in Table 3.6, mainly due to the improvement of K-means output.

The clustering algorithms behaviours with each CQI is very similar to before, and there are not significant differences when data sets with outliers are evaluated separately. A minor difference is that K-means, when used with C-index, performs much better than the other clustering algorithms. Proportionately, the other differences among the three clustering algorithms are negligible.

Regardless the appearance of outliers, ARI again outputs high outcomes (even higher than in Table 3.6 for hierarchical and EM-based algorithms). It means that these algorithms are able to find the correct cluster structures very often. Nevertheless, the internal CQIs behaviours differ compared to results in Table 3.6. Although ARI results are better than before for hierarchical and EM-based algorithms, only C-index for all algorithms and Calinski for K-means and hierarchical algorithms obtain better results for data sets with outliers. Silhouette, DB, and Gamma clearly obtain worse results than in Table 3.6. This is noteworthy because although the correct cluster structures are found, based on ARI, some internal indices are not able to estimate the correct number of clusters.

Indices behaviours in data sets with noise

Results for the 96 data sets with 1 and 2 noisy dimensions are shown in Table 3.4. One of the first conclusions that emerges from these results is the balanced values regardless the used clustering algorithm. In Table 3.3, indices like Silhouette or C-index obtained very different results depending on the used clustering algorithm. However, results for data sets with noise are much more balanced as can be seen in Table 3.4.

Another interesting conclusion is that ARI results are slightly worse than the obtained in Table 3.6 with hierarchical and EM-based clustering algorithms. It means that noisy dimensions affect the clustering algorithms for finding the correct cluster structures. However, only C-index and Calinski with K-means and hierarchical clustering algorithms obtain worse results than the obtained in Table 3.6. Because of these results, C-index is the lowest ranked internal CQI in this comparison. On the other hand, Silhouette, DB, and Gamma clearly outperform previous results even when the correct cluster structures are worse identified according to ARI results. Specifically, Gamma, which is the index with the poorest results

		K					
Index	Algorithm	2	3	4	5	Total	%
Silhouette	K-means	3	7	8	16	34	35.417
	Hierarchical	9	14	14	16	53	55.208
	EM based	6	7	10	18	41	42.708
Calinski	K-means	18	19	11	21	69	71.875
	Hierarchical	18	18	13	20	69	71.875
	EM based	17	14	13	17	61	63.542
C-index	K-means	12	11	9	17	49	51.042
	Hierarchical	3	6	4	21	34	35.417
	EM based	0	1	2	22	25	26.042
DB	K-means	2	6	9	16	33	34.375
	Hierarchical	7	9	13	13	42	43.750
	EM based	18	6	9	5	38	39.583
Gamma	K-means	2	5	5	13	25	26.042
	Hierarchical	3	6	6	15	30	31.250
	EM based	1	2	5	18	26	27.083
ARI	K-means	22	22	22	20	86	89.583
	Hierarchical	24	22	23	23	92	95.833
	EM based	22	22	24	24	92	95.833

Table 3.3: Number and percentage of correct decisions for each CQI, clustering algorithm, and number of clusters K (from 2 to 5) in 96 data sets with 5% and 10% of outliers.

in data sets with outliers, achieves the best results in data sets with noise.

CQI evolution depending on the data characteristics

Previous conclusions about the evolution of each index depending on changes in data are studied now. Data change from "clean" clusters to data with outliers and noisy dimensions. This may lead to conclusions about how these new data characteristics affect the behaviour of CQIs and determine when a particular combination of CQI and clustering algorithm is better.

The complete evolution is shown in Figure 3.1. One of the most interesting findings is that, according to these results, C-index performs worse with *clear* data sets than with outliers. Besides, when Calinski is used to find the correct number of clusters in data sets with 5% outliers, the outcomes are, at least, as competitive as in clear data sets. One important point for examination here is how the introduction of more outliers or noisy dimensions affect the behaviour of each index. Silhouette performs worse in data sets with outliers than in data sets with noise, but the introduction of the second noisy dimension has a bigger effect than the switch from 5% to 10% outliers. This situation is even more marked using Calinski, since the performance substantially decreases compared to other data sets when the second noisy dimension is introduced. DB and Gamma behave similarly: results for data sets with outliers are very poor, whereas values for data sets with one noisy dimension are more competitive

Index	Algorithm	K				Total	%
		2	3	4	5		
Silhouette	K-means	20	19	13	11	63	65.625
	Hierarchical	21	19	13	10	63	65.625
	EM based	24	20	14	9	67	69.792
Calinski	K-means	24	15	12	13	64	66.667
	Hierarchical	24	15	11	12	62	64.583
	EM based	24	16	15	11	66	68.750
C-index	K-means	5	1	3	12	21	21.875
	Hierarchical	3	0	2	12	17	17.708
	EM based	3	1	3	14	21	21.875
DB	K-means	18	19	12	9	58	60.417
	Hierarchical	16	19	11	9	55	57.292
	EM based	16	19	10	9	54	56.250
Gamma	K-means	14	22	15	16	67	69.792
	Hierarchical	15	24	15	14	68	70.833
	EM based	15	22	14	16	67	69.792
ARI	K-means	24	24	21	18	87	90.625
	Hierarchical	24	24	20	16	84	87.500
	EM based	24	24	22	17	87	90.625

Table 3.4: Number and percentage of correct decisions for each CQI, clustering algorithm, and number of clusters (K) in 96 data sets with 1 and 2 noisy dimensions.

compared to *clear* data sets. The performance of most indices decreases when the second noisy dimension is introduced. The exception is the C-index discussed above. Although C-index results are generally very low, results with K-means are better when more outliers are introduced and when the second noisy dimension is introduced than with only 5% outliers and one noisy dimension, respectively. Results obtained with C-index are consistent with results obtained with ARI. The obtained results are better when outliers are introduced than when noisy dimensions are included into the feature set. Assuming that ARI results are the most accurate results because they are obtained by directly comparing partitions instead of internal data structures, Silhouette, Calinski (except for *noi2*), DB, and Gamma obtain results contrary to expectations.

Regarding the algorithms, EM and hierarchical clustering algorithms achieve the best results for clear data sets. For data sets with outliers, hierarchical clustering algorithm achieves results that are, at least, as competitive as the obtained by other clustering algorithms, based on all the internal CQIs, except for C-index, where K-means was the clear winner. In data sets with noisy dimensions, there is not a very clear pattern for clustering algorithms behaviours. The top clustering algorithm changes depending on the used CQI and if one or two noisy dimensions are introduced into the feature set.

Besides the number of correct decisions on the number of clusters decisions, another factor for evaluation is how the value of each index changes depending on data characteristics. The mean values of each CQI for each data situation are shown in Figure 3.2. The aim

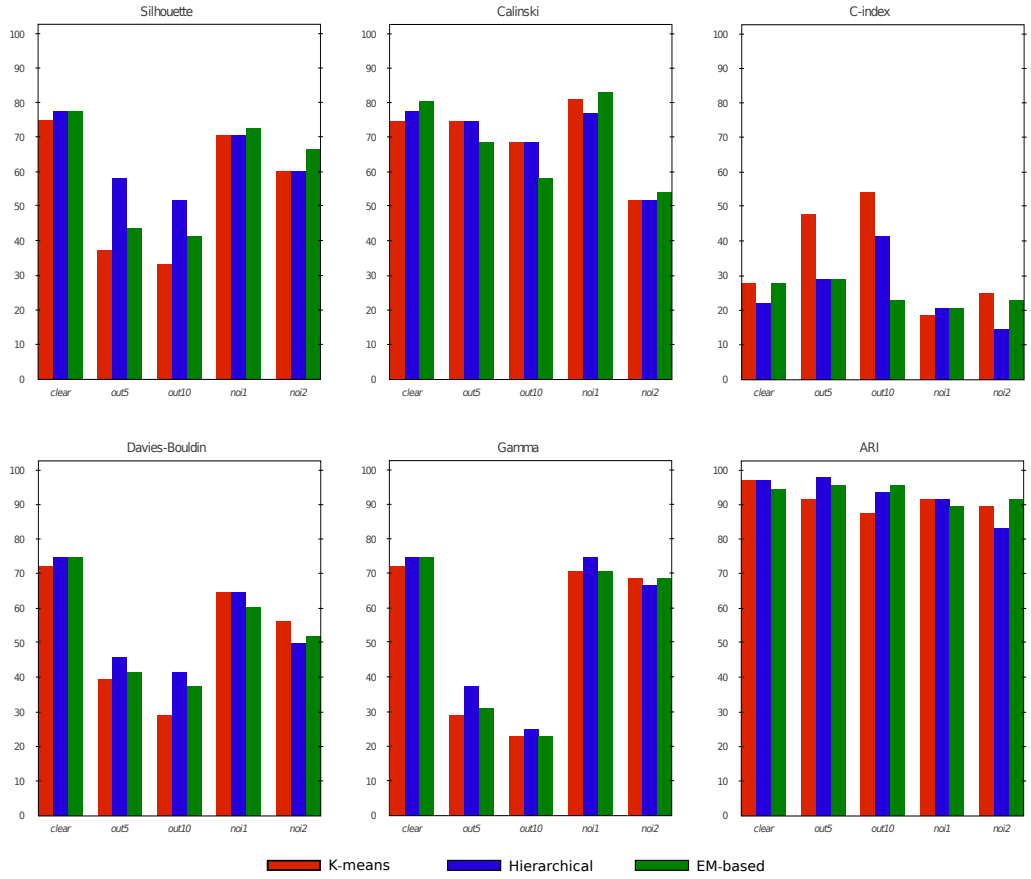


Figure 3.1: Evolution of percentage of correct number of clusters decisions for each CQI, depending on data characteristics and clustering algorithms.

when using Silhouette, Calinski, Gamma, and ARI is to maximize their values, whereas the lower the C-index and DB values, the better. In general, the addition of noisy dimensions particularly affect all indices values, and performances are worse than when *clear* data are evaluated. C-index is again an exception, because results for *clear* data sets are worse than for data sets with outliers. Note also that, when outliers were previously introduced, C-index returned a higher percentage of correct decisions. Here again, when index value is observed, the outliers introduction improves the C-index behaviour. Regarding how the introduction of more outliers or more noisy dimensions affect the output values, Calinski and Gamma performances considerably decrease when the second noisy dimension is introduced. In general, the addition of 5% outliers affects all indices performance except for C-index and ARI.

Random groups and clustering algorithms

After using different clustering algorithms to obtain clustering partitions, a different approach is applied. Data are randomly partitioned to compare CQIs behaviours to previous results

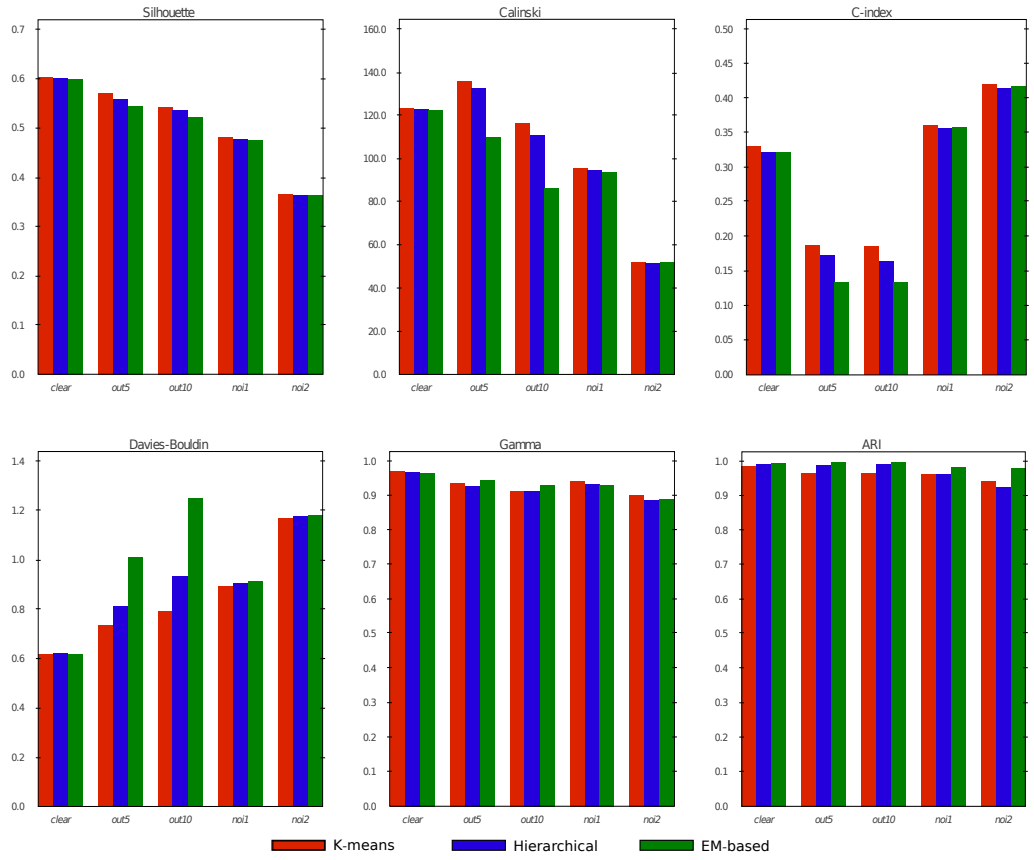


Figure 3.2: Mean values of each CQI for each data type. Note that scale differs depending on the index.

when clustering algorithms were used. The number of random partitions is 50000 for each data set, outputting the same number of quality assessments for each CQI analyzed. This is a random approach, the specified number of repetitions is then considered in an attempt to achieve statistical significance.

In this case, the external CQI is not used to assess how similar random partitions are compared to the original groups. The aim of evaluating random partitions with the internal CQIs is to compare these assessments with evaluations using clustering algorithms, obtaining the random executions percentages that score better values than clustering algorithm validations.

Table 3.5 shows these results, being Silhouette and Gamma the two more logical indices. A logical index is when partitioning evaluation results of clustering algorithms are not usually (less than 0.66% times) outperformed by random partitioning results. The threshold for considering an index as logical is fixed depending on the highest percentage obtained with the two best indices.

On the other hand, random partitions assessed with Calinski, C-index, and DB indices score very different percentages depending on the used data characteristics and clustering

		Data				
Index	Algorithm	*clear*	*out5*	*out10*	*noi1*	*noi2*
Silhouette	K-means	0.21	0.39	0.45	0.49	0.66
	Hierarchical	0.21	0.41	0.47	0.49	0.66
	EM based	0.21	0.41	0.44	0.49	0.66
Calinski	K-means	8.23	6.78	5.77	10.92	0.61
	Hierarchical	7.32	6.28	4.91	11.13	0.66
	EM based	8.91	6.32	6.48	9.82	0.64
C-index	K-means	8.58	13.17	11.20	9.17	4.58
	Hierarchical	6.42	5.25	2.66	7.61	4.27
	EM based	7.95	0.59	3.21	7.72	5.41
DB	K-means	0.36	1.21	1.80	3.12	4.10
	Hierarchical	0.35	2.65	4.69	3.37	4.16
	EM based	0.19	5.41	8.85	3.30	4.31
Gamma	K-means	0.15	0.23	0.22	0.25	0.24
	Hierarchical	0.15	0.23	0.22	0.25	0.24
	EM based	0.15	0.23	0.22	0.25	0.24

Table 3.5: Percentage of cases in which internal CQIs output better evaluation scores for random partitioning than for clustering algorithms.

algorithms. For Calinski, when *noi2* data sets are partitioned, clustering algorithms are beaten only by a maximum of 0.66% random executions. In all other data sets, this percentage was significantly greater, ranging from 4.91% random partitions, which beat the score with hierarchical clustering algorithm in *out10* data, to 11.13% by using the same algorithm to partition *noi1* data.

In the case of C-index, when this index evaluates partitions output with K-means, results are beaten by random partitioning from a 4.58% of cases in *noi2* data to 13.17% in *out5* data. When *out5* data sets are partitioned with EM-based clustering algorithm, the percentage of cases in which random partitioning algorithm scores a better value for C-index was 0.59%; in all other cases using EM or hierarchical clustering algorithms, this value increases up to a maximum of 7.95%.

Random partitioning evaluated with DB index outperforms clustering algorithms in fewer than 1% of the cases for *clear* data, but again this value increases up to 8.85% for *out10* data when EM-based clustering algorithm is used to clustering the data.

Another interesting result is the clear tendency of each internal CQI to choose an "extreme" (minimum or maximum value within the compared range) number of clusters as correct. In the case of Silhouette and C-index, this number of clusters is 2 (which is the minimum number of clusters used in this study). On the other hand, Calinski and DB tends to choose 5 clusters, which is the maximum number of clusters. Gamma is the only index that is not so biased to a set number of clusters. This is important because results may be very biased depending on the selected range of minimum and maximum number of clusters.

3.4 Summary and discussion

This chapter has shown an experimental research about some of the most used CQIs: Silhouette, Calinski, C-index, DB, and Gamma, throwing some light in their behaviours when they are used to validate data with different characteristics, such as having outliers or noisy dimensions. Besides, algorithms from three different clustering paradigms (partitional, hierarchical, and model-based clustering) are used, to also cover a range of some of the most widely used clustering algorithms. The conclusions presented here for each index are related to a previous work, presented by Milligan and Cooper [193], which is an interesting starting point to study many internal indices.

C-index was a measure with a weird behaviour since it achieved better results at evaluating data sets with outliers than data sets without outliers or noisy dimensions. C-index was ranked in third position (in a ranking of 30 measures) by Milligan and Cooper, whereas in this new study was the index with the lowest accuracy. The main reason for this result was the low performance of C-index when it was used for validating clusters with noisy dimensions. The final problem of C-index was that it measured better results for random partitions than for partitions coming from clustering algorithms in a significant percentage of cases.

Another measure that might be placed in a low position of a ranking, according to this new study, is DB. The general behaviour of DB was not as good as other measures, and it got even worse when it was used for validating clustering solutions of data with outliers. This measure was ranked tenth in [193].

Calinski was ranked in first position by Milligan and Cooper. This good position was also demonstrated in this study according to the obtained results. However, Calinski failed in the random partitioning experiment. Its behaviour when results of random partitions were compared to results of clustering algorithms was not as good as that obtained by other indices. It may be an enough reason to not choose Calinski to validate future clustering results.

Silhouette was not ranked by Milligan and Cooper but this index was chosen for this comparison because is widely used to validate clustering solutions. The general results of Silhouette were similar to those obtained by Calinski, obtaining acceptable results in all compared situations and, moreover, demonstrating a higher stability than Calinski throughout the random partitioning experiment.

Finally, Gamma index, which was ranked in fourth position by Milligan and Cooper, behaved again as a weird measure. It was not biased by using a concrete clustering algorithm and demonstrated good stability against random partitions. The only disadvantage of Gamma appeared when outliers were introduced in data, since there was a sharp decrease in its performance. This fact, when compared to Calinski or Silhouette, may be enough reason to choose one of the other indices. Nevertheless, Gamma had a main advantage when compared with all the other indices: it was not biased by the minimum or maximum number of clusters used as range. It means that other measures were biased to choose one of those numbers of clusters, while Gamma did not have this possible disadvantage.

Although these conclusions are related to both the created scenarios and the used algorithms, and, therefore, they cannot be considered as categorical statements, Table 3.6 shows a particular ranking for these indices. Each index is scored for each part of the study according to a value from 1 to 5 depending on the obtained results for each created scenario. According to this ranking, Gamma index should be placed in first position, followed by Silhouette and Calinski. DB would be located in fourth position while C-index would be the worst compared index. However, rather than according to the total results values, each index should be used depending on the intuition about new scenarios, since although Gamma obtained the highest total result, it should not be used to validate results of data with outliers.

Index	General behaviour	Outliers effect	Noise effect	Random stability	Bias to min/max number of clusters	Total values	Final ranking
Silhouette	3	3	4	5	2	17	**2nd**
Calinski	4	4	4	2	2	16	**3rd**
C-index	2	3	1	2	2	10	**5th**
DB	3	2	3	3	3	14	**4th**
Gamma	3	2	4	5	4	18	**1st**

Table 3.6: The different studied parameters are scored for each index on a scale from 1 to 5, where 5 is the best. Values were determined after dividing numeric results into 5 bins. In case of random stability and bias to min/max number of clusters, values were subjectively estimated depending on the conclusions.

Considering the behaviour of each index depending on the different clustering algorithms, it is difficult to find clear patterns indicating the best algorithm-index combination. In general, hierarchical clustering algorithm returned more promising results than K-means or the EM algorithm, but this should not be seen as a categorical conclusion because differences also depended on the characteristics of each data set.

Chapter 4

Semi-supervised subspace hard clustering

4.1 Introduction

The first approach created in this thesis to tackle the semi-supervised subspace clustering problem is based on hard clustering. This approach is called knowledge mapping framework (KMF) and receives partially labeled data as input. The main idea of KMF is to identify subspaces according to the available data labels by using feature subset selection and supervised classification techniques. A clustering algorithm is then used to partition the data into the most interesting clusters depending on the used subspaces and the available labels. According to Section 2.5.1, KMF can be considered a projected clustering algorithm since each instance belongs to a single cluster at the end of this proposal.

KMF is not a final algorithm but a general framework. Several decisions must be made and different algorithms can be used to output the solution depending on the user. A specific instantiation, based on standard decisions and available and known algorithms, is presented here in order to assess the viability of the framework general idea. Both the general framework and the specific instantiation are detailed throughout this chapter (see also [115]).

Chapter outline

The chapter continues in the next section with the notation used throughout this chapter and the details about KMF. The specific instantiation created for this thesis is then presented in Section 4.3. The experimental results are shown in Section 4.4 and, finally, a summary of the proposal and some discussion can be found in Section 4.5.

4.2 Knowledge mapping framework (KMF)

The notation for this proposal follows. Given $\mathcal{X}^N = \mathcal{X}^L \cup \mathcal{X}^U$ a data set of N instances, the main goal is to find a clustering solution for $\mathcal{X}^N$. Specifically, the L instances of $\mathcal{X}^L$ are

described by a feature vector $\mathbf{x} \in \Re^F$ and by a label from a class variable $\mathcal{C} \in \{1, \ldots, C\}$. Therefore $\mathcal{X}^L = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(L)}, c^{(L)})\}$. On the other hand, the $U = N - L$ instances of $\mathcal{X}^U$ are described only by a feature vector $\mathbf{x} \in \Re^F$, and then $\mathcal{X}^U = \{\mathbf{x}^{(L+1)}, \ldots, \mathbf{x}^{(N)}\}$. As commented before, the aim is to find a clustering solution that assigns a label for each instance in $\mathcal{X}^N$. The clustering solution will be led by different subspaces. These subspaces are identified to describe the instances that belong to the known classes and to describe also the instances that do not fit with the known classes, allowing the discovering of new clusters. Hence, the number of final data classes or clusters will be $K \geq C$. Instances in $\mathcal{X}^L$ can vary their labels at the end of the algorithm.

Generally speaking, the KMF framework can be divided into two phases, as follows:

Phase 4.1. Mapping knowledge. Considering the labeled instances ($\mathcal{X}^L$) as the available knowledge, new knowledge in the shape of subspaces is identified by discriminating each known class from the others. Thus, the instance-level knowledge (instances with known class labels) is mapped to feature-level knowledge by finding the respective feature subspaces. Data are then partitioned into different clusters according to the found subspaces. A clustering solution is obtained for each subspace, but only one cluster is finally selected from each solution. The next four steps elaborate on Phase 4.1.

Phase 4.1 Mapping knowledge. For each known class c, a subspace of features (F_c) is found and used to partition the data set. Thus, a data partition (G_c) is found according to each F_c. From each G_c, a genuine cluster (gc_c) is chosen. Therefore, a set of genuine clusters ($\mathcal{GC}$) is obtained from all the known classes. Due to KMF is based on hard clustering, an instance belongs to only one genuine cluster. For this reason, a post-process filter is also applied to select the best cluster for each instance if necessary.

$\mathcal{GC} = \emptyset$
for $c = 1$ **to** C **do**
  Step 4.1.1. $\mathcal{X}_c^L = \text{generateNewDataSet}(\mathcal{X}^L, c)$
  Step 4.1.2. $F_c = \text{supervisedFSS}(\mathcal{X}_c^L)$
  Step 4.1.3. $G_c = \text{clustering}(\mathcal{X}^N, F_c)$
    $gc_c = \text{selectGenuineCluster}(G_c)$
    $\text{add}(\mathcal{GC}, gc_c)$
end for
Step 4.1.4. $\mathcal{GC} = \text{filterRepeated}(\mathcal{GC})$

Step 4.1.1. Phase 4.1 begins by separating, for each known class, the labeled instances in $\mathcal{X}^L$ that belong to class c from the instance subset that belongs to the other known classes. With this, a new binary data set ($\mathcal{X}_c^L$ with $\mathcal{C}_c \in \{0, 1\}$) is generated involving two types of instances, both with known classes, c (coded as 1) and $\neg c$ (coded as 0).

Step 4.1.2. Each new binary data set can be seen as a binary supervised classification problem. Thus, for each known class c, the subset of features (F_c) that gets the most

accurate classification of the binary problem, based on FSS supervised classification techniques, is obtained. This subset of features will be a subspace in which finding a clustering solution in the next step. Figure 4.1 shows this and the previous step for all known classes.

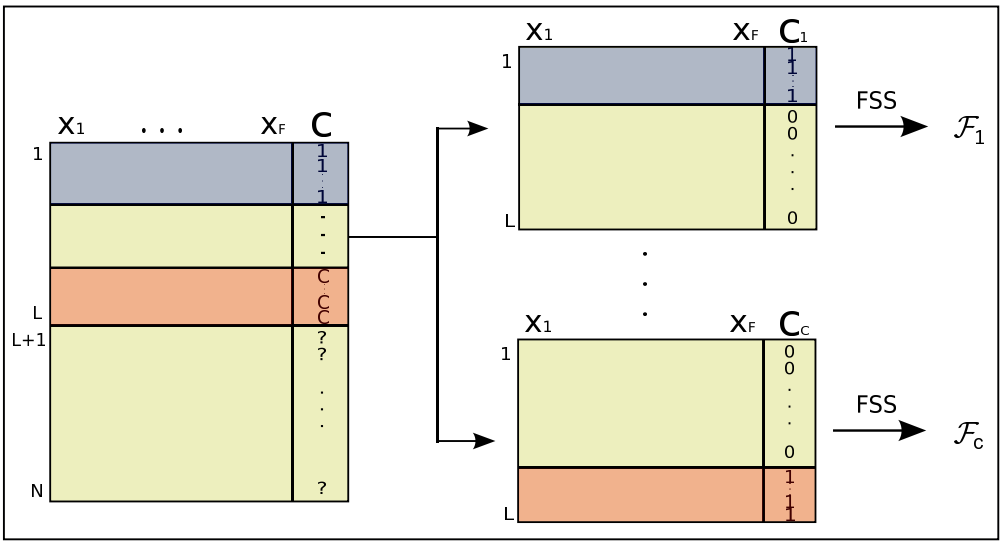


Figure 4.1: Steps 4.1.1 and 4.1.2 of Phase 4.1. The instances subset that belongs to each known class is distinguished from the instances subset that belongs to the remaining known classes. Then the subsets of features that best fit each of these divisions are found.

Step 4.1.3. Once the new feature-level knowledge (subspaces) has been found, it is used to find clustering solutions. This is a discovery step whose aim is to find data partitions depending on the identified subspaces. Thus, a clustering solution (G_c) is obtained for each known class c. All data ($\mathcal{X}^N$) are partitioned by using a clustering algorithm and the previously identified subspace F_c. From G_c, only one cluster, named genuine cluster gc_c, is noteworthy for the final solution, and is the cluster that contains most of the instances of $\mathcal{X}^L$ originally labeled as c. Note this genuine cluster could also be composed of instances of $\mathcal{X}^{\mathcal{L}}$ with a label c', with $c' \neq c$, and also of originally unlabeled instances from $\mathcal{X}^U$.

As mentioned in each step, these three steps are repeated for each known class. Thus, C binary data sets are generated by separating instances in $\mathcal{X}^L$ that belong to each particular class from the remaining labeled instances. C subspaces are then identified by using FSS for each binary supervised classification problem. Finally, C clustering algorithm runs are performed, obtaining C clustering solutions, one for each identified subspace. A set of genuine clusters, $\mathcal{GC} = \{gc_1, \ldots, gc_C\}$, is found from the clustering solutions before beginning Step 4.1.4.

Step 4.1.4. Once C genuine clusters have been extracted from $\mathcal{GC}$, some instances might belong to more than one of these genuine clusters. This approach is based on

hard clustering, i.e., each instance belongs to only one cluster. Therefore, for those instances that belong to more than one cluster, the framework has to select only one cluster for each. Assuming that an instance $\mathbf{x}^{(i)}$ belongs to two genuine clusters gc_c and $gc_{c'}$, if $\mathbf{x}^{(i)} \in \mathcal{X}^L$ and $c^{(i)} = c$, instance $\mathbf{x}^{(i)}$ then remains in gc_c and is deleted from $gc_{c'}$. The same applies vice versa if $c^{(i)} = c'$. On the other hand, if $\mathbf{x}^{(i)} \in \mathcal{X}^L$ and $c^{(i)} \neq c$ and $c^{(i)} \neq c'$, or if $\mathbf{x}^{(i)} \in \mathcal{X}^U$, the distance from $\mathbf{x}^{(i)}$ to the centroids of gc_c and $gc_{c'}$ is calculated. Note that each gc is described using a different subspace of features. Thus, distances are also normalized by dividing by the number of features of each subspace using normalized data. $\mathbf{x}^{(i)}$ remains in the genuine cluster whose distance to its centroid is the minimum after this calculation. Figure 4.2 shows Steps 4.1.3 and 4.1.4 for all known classes.

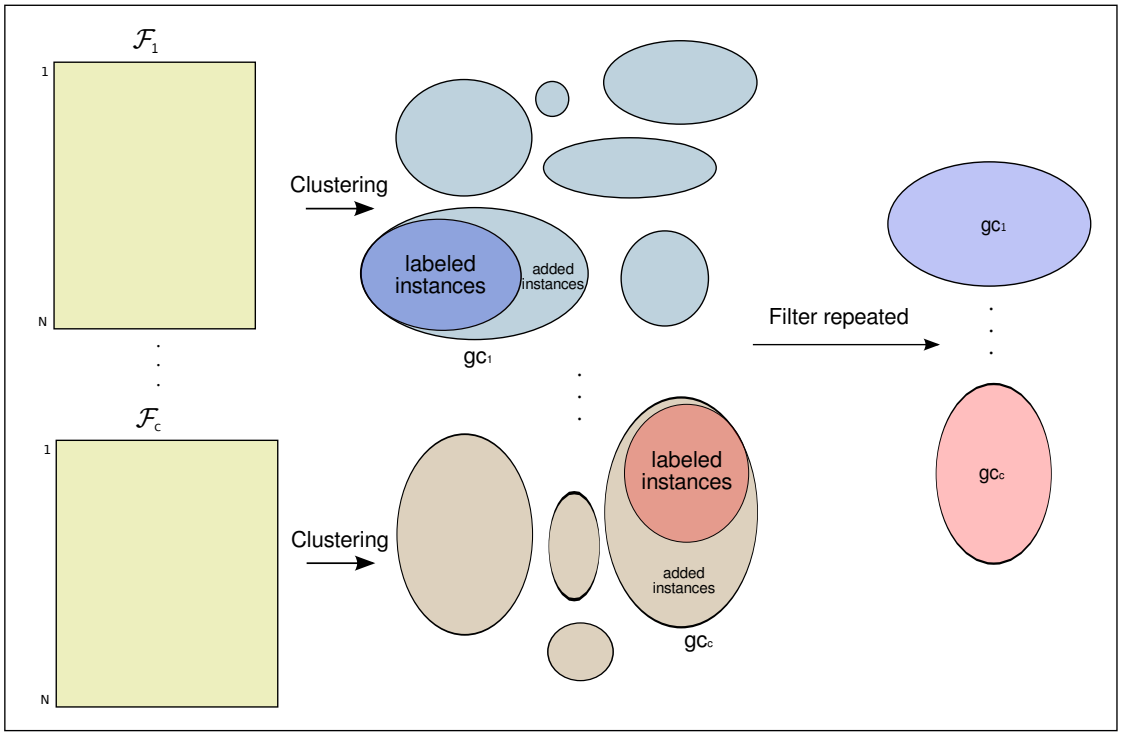


Figure 4.2: Steps 4.1.3 and 4.1.4 of Phase 4.1. For each subspace identified in Step 4.1.2, data are partitioned by using a clustering algorithm. A filtering step deletes repeated instances in genuine clusters before moving on to Phase 4.2.

The partition output at the end of Step 4.1.4 of Phase 4.1 is the final solution if all instances ($\mathcal{X}^N$) are grouped into genuine clusters. If some instances have not been added to $\mathcal{GC}$ by the end of this first phase, Phase 4.2 starts. Any instance of $\mathcal{X}^L$ and $\mathcal{X}^U$ may remain unclassified before Phase 4.2.

Phase 4.2. Discovering new knowledge. This phase aims to discover new clusters containing the instances ($\mathcal{X}^R$) that have not been previously grouped into any genuine cluster. For this purpose, a new subspace (F_{C+1}) is identified and used to partition $\mathcal{X}^R$. This is shown using pseudo-code in Phase 4.2. The three steps below elaborate

on this phase.

Phase 4.2 Discovering new knowledge. If the number of instances grouped into genuine clusters ($NI(\mathcal{GC})$) is fewer than N, then the previously ungrouped $R = N - NI(\mathcal{GC})$ instances are clustered into G_{C+1}, using a clustering algorithm and a new identified subspace F_{C+1}.

if $NI(\mathcal{GC}) < N$ **then**
  Step 4.2.1. $\mathcal{X}^N_{C+1} = \text{generateNewDataSet}(\mathcal{X}^N, C+1)$
  Step 4.2.2. $F_{C+1} = \text{supervisedFSS}(\mathcal{X}^N_{C+1})$
  Step 4.2.3. $G_{C+1} = \text{clustering}(\mathcal{X}^R, F_{C+1})$
end if
return $\mathcal{GC} \cup G_{C+1}$

Step 4.2.1. There are two different subsets of instances at the beginning of this phase: instances grouped into genuine clusters ($\mathcal{X}^T$), and instances that do not belong to $\mathcal{GC}$ ($\mathcal{X}^R$), with $\mathcal{X}^T \cup \mathcal{X}^R = \mathcal{X}^N$ and $\mathcal{X}^T \cap \mathcal{X}^R = \emptyset$. These two subsets generate a new binary data set, $\mathcal{X}^N_{C+1}$, with $\mathcal{C}_{C+1} \in \{0, 1\}$.

Step 4.2.2. A subspace F_{C+1} is identified from $\mathcal{X}^N_{C+1}$ by using FSS for the created binary supervised classification problem, as in Step 4.1.2 of Phase 4.1. This new subspace discriminates instances whose classification was already found in Phase 4.1 from instances that must be classified in this phase.

Step 4.2.3. Then, a clustering algorithm is used to partition $\mathcal{X}^R$ in F_{C+1}, obtaining a new clustering solution with C' new clusters.

At the end of this second phase, the final solution is reached, and the final instances classification is $\mathcal{GC} \cup G_{C+1}$, with all instances grouped either in one of the C genuine clusters or in a new cluster. Figure 4.3 shows the process for this phase.

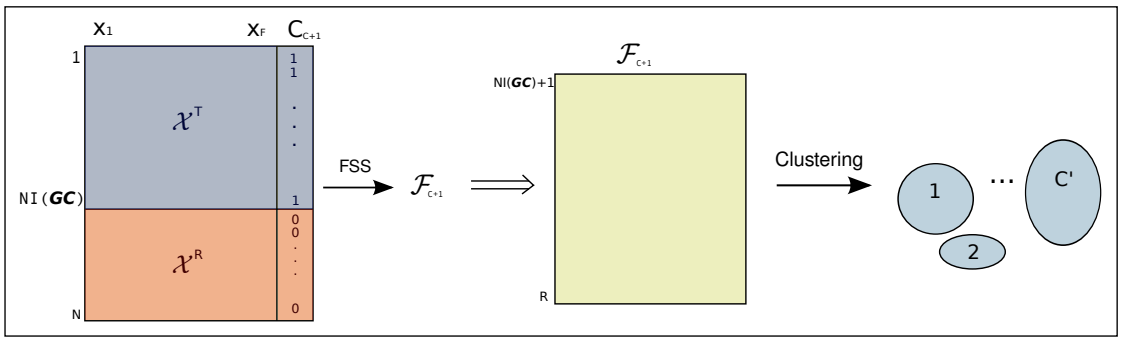


Figure 4.3: Phase 4.2. A clustering algorithm is again applied to the last subspace of features identified to discover new clusters.

To sum up, the original instance-level knowledge is used to obtain feature-level knowledge, represented in different subspaces. These subspaces are used to partition the data into

different clusters. At the end of the process, if Phase 4.2 is necessary as expected, there will be $C + 1$ subspaces characterizing K clusters, where $K = C + C'$, with C' the number of clusters found in the last phase.

4.3 Knowledge mapping specific instantiation

The proposal explained in the previous section must be instantiated using several supervised and unsupervised algorithms. The decisions made for this instantiation are based on using standard and available algorithms.

Steps 4.1.2 and 4.2.2 are solved by using a wrapper FSS (see Chapter 2.5), where results are evaluated by selecting the subset of features that returns the highest classification accuracy. In this case, logistic regression (see Section 2.2) is used as supervised classification algorithm, and GA (see Chapter 2.5) as search method in the feature space.

A constrained clustering algorithm is chosen for Step 4.1.3 to take advantage of the available class label information. This information can be mapped to pairwise constraints, obtaining must-link constraints if two instances have the same class label and cannot-link constraints if their class labels are different, as explained in Section 2.4.2. The algorithm used in this implementation is MPCKM, previously presented in Section 2.4.2. Besides, a greedy approach based on maximizing the objective function of MPCKM is used to estimate the number of clusters when necessary. This estimation process iterates MPCKM in a range of clusters $\{k_{min}, \ldots, k_{max}\}$, reaching the convergence criterion when $\mathcal{J}_{MPCKM}(k_i) \leq \mathcal{J}_{MPCKM}(k_{i+1})$ holds the first time, returning k_i as the number of clusters, with $k_{min} \leq k_i \leq k_{max}$.

Hierarchical clustering with Ward's method (see Section 2.3.1) is chosen for Step 4.2.3. This approach returns a hierarchy that can be seen as a graphical representation for readily observing the distances between clusters. This, along with its simplicity, were the reasons for its choice.

Finally, there are many internal validation indices in clustering, as commented in Section 2.3.2 and studied in Chapter 3, that can be used to select the final number of clusters. A voting scheme is proposed here, using Silhouette, Calinski, DB, and Gamma indices. All these indices were detailed in Section 3.2. In this scheme, each index votes a number of clusters depending on its obtained values. The number of clusters with the highest number of votes is finally selected. Calinski breaks possible ties because it was the best ranked index by Milligan and Cooper in [193], and also obtained good results in the comparison presented in the previous chapter.

4.4 Experimental results

The instantiation for the proposal (the algorithm from now on) is evaluated using both real and synthetic data sets. Although these data are completely labeled, the full set of labels is used only for validation purpose. A subset of instances maintain the label in each experiment to create the partially labeled data as input for the algorithm. Different percentages of labeled

instances are used to correctly assess the algorithm behaviour when the percentage of available labels changes. The algorithm is evaluated at the end of Phase 4.1, checking whether the identified subspaces are correct, and also at the end of Phase 4.2 (final results) using several external indices to output a complete assessment. Finally, for comparison purposes, the same data sets are evaluated using MPCKM.

Note that MPCKM is also part of the proposed KMF instantiation, since it is used as the clustering algorithm in Step 4.1.3. However, this instantiation is not an extension of MPCKM, since any other algorithm can be used in its place. Therefore, it can be useful to use MPCKM for comparison purposes to check whether KMF is helpful for improving an algorithm like MPCKM.

4.4.1 Data

Seven real data sets from the UCI Machine Learning Repository [91] and three synthetic data sets were used in the experiments. All data sets have similar characteristics: hundreds of instances, tens of continuous features, and several classes. Table 4.1 shows the number of instances, features, and classes of both the real and synthetic data sets.

The synthetic data sets were generated using a clustering generator as described in earlier researches focusing on subspace clustering [143, 200]. In this kind of data, there are data structures hidden in different subspaces. Therefore some features are completely useless for finding some hidden structures, but absolutely necessary for finding others. The difference among the synthetic data sets is the number of dimensions used in each case (15, 25, and 50). The main feature of KMF is to identify different subspaces before the clustering is performed, and this subspace search is mainly evaluated with the synthetic data sets. On the other hand, the chosen real data sets are suitable for the experimentation since they may contain only full-dimensional subspace clusters, and the framework must be also evaluated when data could be described without using different subspaces.

Data	N	F	K
glass	214	9	6
image	2310	19	7
iris	150	4	3
shape	160	17	9
vehicle	842	18	4
vowel	990	10	11
wine	178	13	3
syn15	1461	15	8
syn25	1454	25	8
syn50	1461	50	8

Table 4.1: Number of instances (N), features (F), and classes (K) of seven real from UCI and three synthetic data sets used in the experiments.

All data sets are fully labeled, therefore some instance labels are hidden to get partially labeled data. The full set of labels is used for evaluation. To gain more insight into the algorithm behaviour, four percentages of labeled instances are used: 10%, 20%, 30%, and 40%. The instances that remain labeled are selected completely at random with only one consideration: if a data set has C known classes, instances corresponding to $[\frac{C}{2}]+1$ classes are candidates for random selection as labeled instances, whereas instances from the remaining classes are fixed as completely unknown. This is made to evaluate the algorithm behaviour

not only classifying instances into known classes, but also for discovering new classes. Since selection is at random, 100 executions of the algorithm are run for each percentage of labeled instances and data set, to achieve statistical significance. Each execution contains different labeled instances with high probability.

4.4.2 Evaluation process

It is assumed in the evaluation process that original class labels match natural clusters. This is similar to the *cluster assumption*, commented in Section 2.4.1, which states that highly similar instances must have the same label (or class). The KMF evaluation process is divided into two stages. The first stage checks the genuine clusters behaviour (see Section 4.2) and the second stage assesses the final clusters at the end of the algorithm. The original labels of each data set are used to evaluate the classification obtained by the algorithm in across both stages.

After the genuine clusters have been found, they are evaluated using two indices: *recall* and *precision*, comparing the partition obtained by the genuine clusters with the real class labels. These two indices are used in information retrieval [249] and also in statistical classifications. Recall measures how many instances of a particular class are in the same cluster taking into account those instances of the same class that are not. Precision measures how many instances in a cluster are from a particular class taking into account all the instances in the cluster. In both cases, the range of scores is from 0 to 1, 1 being the best score. More details can be found in Section 2.3.2.

21 indices are used to establish the quality of each partition against the real labels comparing our algorithm with the MPCKM algorithm. The complete list of used indices is: ARI, Hamann, Czekanowski, Kulczynski, McConnaughey, Peirce, Wallace1, Wallace2, Gamma, Sokal1, Fager, Sokal2, Sokal3, Gower, Roger, Kruskal, Pearson, Rand, Jaccard, Folkes, and Russel. But, detailed results for only four indices are reported, as some of the 21 indices are quite similar. The four indices are: ARI, Gamma, Gower, and Russel, and were presented in Section 2.3.2. ARI and Gamma are very known indices which are demanding and balanced, i.e., these indices take into account both good and bad results in partitions. On the other hand, Gower and Russel are chosen as example of indices that output very good and bad results, respectively, due to the evaluation method used. The complete description of the remaining indices can be found in package *Validator* of the *R-project* software [216], or in [7].

4.4.3 Results

Phase 4.1

The genuine clusters evaluation, obtained by comparing the obtained results for those clusters to the available real labels, are shown for real data sets in Table 4.2 and for synthetic data sets in Table 4.3. Recall and precision results show whether the clusters obtained in Phase 4.1 were able to distinguish each known class from the other classes (precision), and, also

		Recall	Precision
Data	%	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd
glass	10	0.42 ± 0.23	0.68 ± 0.19
	20	0.50 ± 0.24	0.71 ± 0.19
	30	0.62 ± 0.24	0.74 ± 0.18
	40	0.70 ± 0.22	0.77 ± 0.18
image	10	0.76 ± 0.22	0.83 ± 0.19
	20	0.77 ± 0.19	0.80 ± 0.02
	30	0.85 ± 0.14	0.84 ± 0.18
	40	0.92 ± 0.08	0.81 ± 0.19
iris	10	0.79 ± 0.26	0.98 ± 0.05
	20	0.90 ± 0.13	0.95 ± 0.09
	30	0.96 ± 0.04	0.94 ± 0.12
	40	0.93 ± 0.07	0.94 ± 0.12
shape	10	0.73 ± 0.23	0.71 ± 0.18
	20	0.81 ± 0.17	0.74 ± 0.17
	30	0.89 ± 0.12	0.75 ± 0.18
	40	0.93 ± 0.09	0.77 ± 0.19
vehicle	10	0.31 ± 0.11	0.68 ± 0.16
	20	0.40 ± 0.00	0.78 ± 0.00
	30	0.50 ± 0.00	0.83 ± 0.00
	40	0.61 ± 0.06	0.88 ± 0.10
vowel	10	0.76 ± 0.22	0.83 ± 0.19
	20	0.77 ± 0.19	0.80 ± 0.02
	30	0.85 ± 0.14	0.84 ± 0.18
	40	0.92 ± 0.08	0.81 ± 0.19
wine	10	0.76 ± 0.22	0.83 ± 0.19
	20	0.77 ± 0.19	0.80 ± 0.02
	30	0.85 ± 0.14	0.84 ± 0.18
	40	0.92 ± 0.08	0.81 ± 0.19

Table 4.2: Average ($\bar{x}$) and standard deviation (sd) recall and precision for genuine clusters in each real data set and percentage (%) of labeled instances. Results were calculated by averaging execution results.

whether the identified subspaces were able to group instances that belonged to originally known classes (recall).

A common behaviour across all real data sets is that recall improved more than precision when the percentage of labeled instances grew (see Table 4.2). Comparing results with 10% and 40% of labeled instances, recall improved almost 24% on average, whereas precision improved just under 7% on average. In any case, the initial results with 10% of labeled instances were higher for precision than for recall in all data sets, except for *shape*. For this reason, taking into account the good initial results for precision, and the high improvement for recall when labels percentage grew, Phase 4.1 results were very competitive even when the importance of subspaces in these real data sets was limited. Specifically, with 40% of labeled instances, recall ranged from 0.61 ± 0.06 for *vehicle* to more than 0.90 on average for

image, *iris*, *shape*, and *wine*. Precision results ranged from 0.77 ± 0.18 and 0.77 ± 0.19 in *glass* and *shape*, respectively, to 0.94 ± 0.12 in *iris*.

		Recall	Precision
Data	%	$\bar{x}$ ± sd	$\bar{x}$ ± sd
syn15	10	0.84 ± 0.20	0.90 ± 0.16
	20	0.87 ± 0.17	0.91 ± 0.15
	30	0.90 ± 0.13	0.92 ± 0.14
	40	0.94 ± 0.10	0.92 ± 0.14
syn25	10	0.92 ± 0.13	0.95 ± 0.10
	20	0.96 ± 0.08	0.96 ± 0.10
	30	0.98 ± 0.05	0.97 ± 0.09
	40	0.99 ± 0.03	0.97 ± 0.09
syn50	10	0.93 ± 0.13	0.94 ± 0.13
	20	0.95 ± 0.10	0.94 ± 0.12
	30	0.97 ± 0.07	0.95 ± 0.12
	40	0.98 ± 0.04	0.95 ± 0.12

Table 4.3: Average ($\bar{x}$) and standard deviation (sd) recall and precision for genuine clusters in each synthetic data set and percentage (%) of labeled instances. Results were calculated by averaging execution results.

As expected, synthetic data results (see Table 4.3) were better on average than real data results, because subspaces were highly important in these cases. Again, there was more improvement for recall than for precision when the percentage of labels grew (around 7% and 2%, respectively). Final results were very accurate, and the genuine clusters classifications for *syn25* and *syn50* with 40% of labeled instances were near perfect.

Based on these results, it can be said that the presented KMF instantiation achieved the first phase goal, i.e., to correctly identify the subspaces that described the known classes, since new instances with the same label (although hidden to the input) were correctly grouped into genuine clusters.

Phase 4.2

After the genuine clusters are found, the remaining final clusters are obtained at the end of Phase 4.2 by identifying a new subspace and using hierarchical clustering. All these clusters are the final clustering solution for each scenario and are compared to real (and partially hidden) labels. The aim is to check the evolution of each result and the algorithm general behaviour depending on the percentage of labeled instances and data characteristics. Results for real and synthetic data sets are listed in Table 4.4 and in Table 4.5, respectively. Taking into account the properties of the indices presented in Section 2.3.2, results must be deeply studied using ARI and Gamma, since Gower and Russel are shown as examples of very radical indices, i.e., indices that evaluate partitions with a bias to high and low values, respectively.

The real data results evolution can be seen in Table 4.4. Behaviours differed depending on the data set, since the improvement with *glass*, *vehicle*, *vowel*, and *wine* was around 24%

on average from 10% to 40% of labeled instances using ARI. Nevertheless, the improvement was only around 7% on average with *image*, *iris*, and *shape*. This also depended on the index, since looking at Gamma results, the improvements were more moderate because initial results (with 10% of labels) were higher using Gamma than ARI.

		ARI	Gamma	Gower	Russel
Data	%	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd
glass	10	0.20 ± 0.04	0.42 ± 0.04	0.83 ± 0.02	0.09 ± 0.03
	20	0.24 ± 0.05	0.44 ± 0.03	0.84 ± 0.02	0.10 ± 0.02
	30	0.31 ± 0.05	0.48 ± 0.03	0.86 ± 0.02	0.11 ± 0.02
	40	0.40 ± 0.06	0.53 ± 0.03	0.88 ± 0.02	0.13 ± 0.02
image	10	0.50 ± 0.09	0.56 ± 0.06	0.93 ± 0.02	0.07 ± 0.01
	20	0.52 ± 0.07	0.57 ± 0.05	0.94 ± 0.01	0.08 ± 0.01
	30	0.56 ± 0.08	0.61 ± 0.06	0.94 ± 0.02	0.09 ± 0.01
	40	0.60 ± 0.07	0.64 ± 0.05	0.94 ± 0.03	0.10 ± 0.01
iris	10	0.63 ± 0.15	0.68 ± 0.11	0.92 ± 0.03	0.20 ± 0.05
	20	0.73 ± 0.08	0.71 ± 0.06	0.94 ± 0.02	0.25 ± 0.01
	30	0.72 ± 0.12	0.75 ± 0.09	0.93 ± 0.03	0.24 ± 0.02
	40	0.69 ± 0.12	0.73 ± 0.09	0.93 ± 0.03	0.23 ± 0.04
shape	10	0.53 ± 0.06	0.57 ± 0.05	0.95 ± 0.01	0.06 ± 0.07
	20	0.54 ± 0.08	0.59 ± 0.06	0.94 ± 0.01	0.07 ± 0.01
	30	0.56 ± 0.07	0.61 ± 0.05	0.95 ± 0.01	0.08 ± 0.02
	40	0.59 ± 0.09	0.63 ± 0.07	0.95 ± 0.01	0.08 ± 0.07
vehicle	10	0.12 ± 0.03	0.41 ± 0.03	0.80 ± 0.03	0.09 ± 0.02
	20	0.16 ± 0.03	0.42 ± 0.02	0.81 ± 0.02	0.09 ± 0.01
	30	0.23 ± 0.03	0.44 ± 0.02	0.84 ± 0.01	0.10 ± 0.01
	40	0.33 ± 0.03	0.49 ± 0.02	0.87 ± 0.01	0.11 ± 0.01
vowel	10	0.09 ± 0.04	0.27 ± 0.01	0.88 ± 0.03	0.02 ± 0.00
	20	0.17 ± 0.02	0.31 ± 0.01	0.91 ± 0.03	0.03 ± 0.00
	30	0.23 ± 0.04	0.34 ± 0.02	0.91 ± 0.02	0.03 ± 0.00
	40	0.33 ± 0.06	0.42 ± 0.04	0.93 ± 0.01	0.04 ± 0.00
wine	10	0.48 ± 0.11	0.59 ± 0.07	0.88 ± 0.03	0.18 ± 0.04
	20	0.55 ± 0.10	0.63 ± 0.08	0.90 ± 0.02	0.20 ± 0.04
	30	0.63 ± 0.10	0.69 ± 0.07	0.92 ± 0.02	0.21 ± 0.04
	40	0.70 ± 0.09	0.75 ± 0.07	0.93 ± 0.02	0.23 ± 0.03

Table 4.4: Real data sets average ($\bar{x}$) and standard deviation (sd) for each external index and different percentages (%) of labeled instances. Results are obtained by comparing the final clustering solutions to the real labels.

Table 4.5 shows the results for the synthetic data sets. *syn15* obtained 0.80 ± 0.10 using ARI with 40% of labeled instances, but this result was higher with *syn50* (0.90 ± 0.07), and even more so with *syn25* (0.92 ± 0.07). As can be seen in the table, good results denoted again the importance of subspaces for the best framework exploitation. Note also the importance of the number of dimensions in data. When the number of dimensions was either too high or low, the FSS search technique may not find the best features subset in some cases.

Therefore, results using KMF when clustering data with subspaces were very good. Nev-

		ARI	Gamma	Gower	Russel
Data	%	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd	$\bar{x} \pm$ sd
syn15	10	0.75 ± 0.12	0.77 ± 0.10	0.97 ± 0.02	0.11 ± 0.02
	20	0.76 ± 0.10	0.78 ± 0.09	0.97 ± 0.01	0.11 ± 0.02
	30	0.77 ± 0.10	0.79 ± 0.09	0.97 ± 0.01	0.12 ± 0.02
	40	0.80 ± 0.10	0.81 ± 0.09	0.97 ± 0.01	0.12 ± 0.02
syn25	10	0.86 ± 0.08	0.87 ± 0.07	0.98 ± 0.01	0.13 ± 0.01
	20	0.89 ± 0.08	0.89 ± 0.07	0.99 ± 0.01	0.13 ± 0.01
	30	0.91 ± 0.07	0.91 ± 0.07	0.99 ± 0.01	0.13 ± 0.01
	40	0.92 ± 0.07	0.93 ± 0.07	0.99 ± 0.01	0.14 ± 0.01
syn50	10	0.83 ± 0.09	0.84 ± 0.08	0.98 ± 0.01	0.12 ± 0.01
	20	0.85 ± 0.08	0.86 ± 0.07	0.98 ± 0.01	0.13 ± 0.01
	30	0.88 ± 0.08	0.89 ± 0.07	0.98 ± 0.01	0.13 ± 0.01
	40	0.90 ± 0.07	0.91 ± 0.06	0.99 ± 0.01	0.13 ± 0.01

Table 4.5: Synthetic data sets average ($\bar{x}$) and standard deviation (sd) for each external index and different percentages (%) of labeled instances. Results are obtained by comparing the final clustering solutions to the real labels.

ertheless, it is necessary to compare these results with a different approach, to obtain a more accurate feedback about the behaviour of KMF. This is even more important when clustering real data, since KMF is not focused on data without subspaces. Therefore, the comparison with a clustering algorithm that is not focused on subspaces may indicate whether the obtained results were competitive.

4.4.4 Comparison with a constrained clustering algorithm

The approach employed for comparison is a much used constrained K-means, previously explained in Section 4.3, and abbreviated as MPCKM. This algorithm iterates to satisfy must-link and cannot-link constraints using an adaptive metric for each cluster depending on the pairwise relations. This algorithm does not compute the number of clusters, which has to be entered as a parameter. MPCKM is run within a range from the number of known classes in each data set to 20, and the number of clusters is selected in the algorithm by minimizing $\mathcal{J}_{MPCKM}$. The above results for synthetic data were very accurate, but the aim with this comparison is to check whether KMF is also competitive when clustering real data sets, although subspaces are not necessarily a primary feature of these data.

Table 4.6 presents the average number of clusters selected for each approach depending on the data set and the percentage of labeled instances. KMF was, in general, more accurate than MPCKM at selecting the number of clusters. This applied mainly to the synthetic data sets: the number of clusters selected by KMF was on average 11 when there were 15 features; but it estimated eight clusters, the real number, when there were 25 and 50 features. The estimation of the number of clusters when clustering real data was not accurate when using KMF, but neither MPCKM was able to select the correct number of clusters. KMF selected on average a number of clusters closer to the real number than MPCKM for *glass*, *iris*,

Data	%	KMF	MPCKM
glass (6)	10	9.50	11.69
	20	9.20	11.59
	30	8.91	11.30
	40	8.84	11.45
image (7)	10	16.40	12.80
	20	17.20	12.10
	30	16.90	11.80
	40	15.56	11.90
iris (3)	10	8.30	8.40
	20	5.40	8.00
	30	6.10	7.70
	40	6.80	7.20
shape (9)	10	12.60	10.70
	20	11.10	9.75
	30	10.10	9.80
	40	10.10	9.60
vehicle (4)	10	6.16	13.34
	20	5.87	12.87
	30	5.90	13.39
	40	5.92	13.95
vowel (11)	10	11.00	14.80
	20	12.50	13.80
	30	16.00	13.30
	40	12.90	13.80
wine (3)	10	5.33	11.92
	20	5.45	11.74
	30	5.48	11.28
	40	5.37	11.35
syn15 (8)	10	11.15	12.13
	20	11.33	12.48
	30	11.05	12.72
	40	11.92	12.68
syn25 (8)	10	8.68	11.63
	20	8.78	11.60
	30	8.82	11.48
	40	8.89	11.57
syn50 (8)	10	8.16	11.29
	20	7.95	11.48
	30	7.91	11.62
	40	7.73	11.99

Table 4.6: Average number of clusters selected for each approach and percentage (%) of labeled instances. Real number of clusters are shown within parenthesis.

vehicle, and *wine*. Note that results related to the estimation of the number of clusters using KMF completely depends on the clustering algorithm that is used during Phase 4.2. For this reason, this results should be taken wisely, since not only different clustering algorithms, but also different indices or processes to evaluate the final selection could be used. Although how each algorithm selected the number of clusters is an interesting feature, it is necessary to compare the final clustering solutions independently of the selected number of clusters.

The Wilcoxon signed-rank test [260], a nonparametric statistical hypothesis test which can reveal the existence of significant differences between two distributions, is used to compare results output by KMF and MPCKM. The null hypothesis is that there are not statistical differences between the two distributions. Therefore, the test checks whether there is significant statistical difference between results from both algorithms, with a significance level of 5%, for different data sets and labeled instances percentages.

The Wilcoxon results for the 21 indices are shown in Table 4.7, and indicate how many indices of each approach were significantly better than for the other, and how many indices did not achieve significant differences. Results were quite similar for the *glass*, *vehicle*, and *wine* real data sets, where MPCKM had a slight advantage. This small difference became even lower depending on the labeled instances percentage, and results for *glass* and 30% of labeled instances, or *vehicle* and 10% were equal. In other cases, like for example, *wine* and 30% of labeled instances, KMF achieved better results than MPCKM. Results were even more similar for *image*, *iris* (except with 20%, where KMF clearly outperformed MPCKM), *shape*, and *vowel*. Therefore, KMF was very competitive when clustering data with no subspaces.

In the case of synthetic data sets, MPCKM performed better in *syn15*. MPCKM achieved better results in *syn50* with 10% and 20% of available labels, showing that the proposal was not able to find the correct subspaces with a low percentage of labels, but in this data set, KMF clearly outperformed MPCKM with 30% and 40%. KMF also obtained better results in *syn25* regardless the percentage of labeled instances. As previously mentioned, the number of features is one of the most important data characteristic for KMF, because either a small or a large number of features could lead to poor decisions, depending on the FSS technique.

As a summary, KMF obtained competitive results compared to MPCKM when clustering data that do not need to be described by using different subspaces. On the other hand, if the presence of subspaces are a characteristic of the target data, KMF outperformed MPCKM, provided there was a large enough number of features (at least 25 features based on this experimentation), or when the percentage of labeled instances was high ($\geq$ 30%).

4.5 Summary and discussion

This chapter has presented a framework, called KMF, that, having available class label information for some instances, aims to clustering data using different identified subspaces. This goal is achieved by mapping the instance-level knowledge to feature-level knowledge, and taking advantage of supervised methods to output this information. The search for different subspaces not only outputs more parsimonious clustering solutions that can be easily inter-

preted, but also is able to find data structures that might remain hidden without searching for subspaces.

After the framework presentation, a specific instantiation is proposed. This instantiation is based on using very known and available algorithms, trying to assess the framework ideas. The main reason for using known algorithms was the easy understanding of the framework. Besides, using this kind of algorithms allowed to validate the instantiation quickly. On the other hand, results showed some limitations in determined situations that could be overcome using other kind of algorithms. The proposal also automatically selects the number of clusters, which is a typical problem in all tasks related to clustering, by using some indices for validating the clustering solutions.

Some conclusions can be drawn from the creation of this proposal and the achieved results. KMF identifies subspaces completely based on the available data labels. These subspaces are then used as different dimensions where data partitions are found. Regarding the known labels, there are two noteworthy conclusions. First of all, the available data labels can be used to guide the clustering process. This is the main conclusion for most semi-supervised clustering algorithms. Related to the presented approach, some semi-supervised clustering algorithms used the known information about data labels or pairwise relations to constraint the space search or to learn a new metric, as commented in Section 2.4.2. In this case, data labels guide the search for subspaces, and the clustering process completely depends on these subspaces. The second conclusion is directly drawn from the previous one. Since data labels are used as a starting point of KMF, they must be completely reliable to obtain accurate results. This is a known trouble for this kind of problems and is preferable to have a lower amount of reliable information that a higher amount of less accurate information.

Also related to the search for subspaces, another important data characteristic for KMF is the number of features. An FSS for a supervised classification problem is used here for finding each subspace. According to the obtained results, the FSS process is very accurate for data described with 25 features. The combination of the number of features and the available data labels is also important, since KMF obtained very different results for a data set with 50 features depending on the percentage of available data labels. This is possibly related to whether data are balanced or not. When the number of labeled instances for a specific class is very different from the number of labeled instances for all the other known classes, the FSS process is performed in an unbalanced data set. The binary supervised classification problem becomes more difficult for this kind of data and, therefore, the FSS process may become less accurate. Assuming the commented limitations and depending on the scenario, KMF obtained accurate results when clustering synthetic data sets. The main characteristic of these data is that clusters are characterized in different subspaces. Thus, the clustering solutions for the target data of KMF achieved the goal of improving results when compared to a state-of-the-art semi-supervised clustering algorithm.

On the other hand, KMF was also validated with real data sets. These data are not characterized by different subspaces and, therefore, the search for subspaces of KMF may not help to obtain accurate clustering results in this case. Nevertheless, the results obtained

Data	%	KMF	MPCKM	NoDiff
glass	10	4	12	5
	20	7	12	2
	30	9	9	3
	40	4	10	7
image	10	0	0	21
	20	0	0	21
	30	0	0	21
	40	0	0	21
iris	10	0	0	21
	20	18	1	2
	30	3	1	17
	40	0	2	19
shape	10	0	1	20
	20	0	0	21
	30	0	0	21
	40	0	4	17
vehicle	10	10	11	0
	20	9	11	1
	30	9	12	0
	40	7	14	0
vowel	10	2	17	2
	20	0	4	17
	30	1	6	14
	40	0	2	19
wine	10	9	11	1
	20	8	10	3
	30	10	7	4
	40	4	15	2
syn15	10	2	19	0
	20	2	19	0
	30	2	19	0
	40	2	19	0
syn25	10	17	4	0
	20	17	4	0
	30	17	2	2
	40	19	1	1
syn50	10	2	19	0
	20	2	19	0
	30	11	4	6
	40	17	4	0

Table 4.7: Number of external indices out of 21 returning no statistical difference between the approaches (NoDiff), or, number of times one approach outperforms the other for each percentage of knowledge and data set.

with KMF were competitive, depending on the data, when compared to a semi-supervised clustering algorithm that does not search for subspaces. The important conclusion about these results is that, although KMF searches for subspaces regardless of the input data, the FSS process should not affect very much the results if subspaces cannot be found. The most extreme case is whether the FSS process identifies always the same subspace for all known classes, i.e., there is only a feature subset which is suitable to distinguish between each known class and the others. Note that the clustering step will be performed with always the same subspace in that case, but different genuine clusters will be identified. Although this is a desirable situation for data which are not described by different subspaces, KMF is completely exploited when different subspaces can be identified.

Chapter 5

Semi-supervised subspace soft clustering

5.1 Introduction

The second proposal, called semi-supervised subspace model-based clustering (3SMBC), is based on probabilistic (and soft) clustering and adapts model-based clustering approach, presented in Section 2.3.1, integrating available instance labels and subspace search together with the EM algorithm to obtain the final clustering solution. Therefore, as for the previous proposal, the 3SMBC input data must be partially labeled data.

Besides the soft clustering instead of the hard clustering approach, the main difference between 3SMBC and KMF is that the former is a closed-algorithm and not a framework that can be instantiated using different algorithms. Another important difference is that the available information was used to find subspaces in KMF, and these subspaces were then used to find the clustering solution. However, subspaces and clusters are simultaneously searched in 3SMBC.

This approach is based on the traditional concept of finite mixture models, where an instance is assumed to be generated by a probabilistic model given by a finite mixture of distributions [29]. 3SMBC work is described throughout this chapter (see also [112]).

Chapter outline

The notation for this chapter is presented in the next section together with the basic theory of mixture modeling. It is used as basis to detail the proposal about semi-supervised subspace soft clustering in Section 5.3. The chapter continues in Section 5.4 showing the experimental results using both synthetic and real data. A comparison with related algorithms is also shown in that section. Finally, Section 5.5 covers a summary of the proposal and some discussion about it.

5.2 Basic theory

The underlying theory of mixture modeling is introduced in this section as the groundwork for 3SMBC. First of all, the notation in this chapter is similar to that used in the previous chapter. Let $\mathcal{X}^N = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ be a partially labeled data set of instances described by continuous features in a space of dimension F, that is, $\mathbf{x}^{(i)} \in \Re^F, \forall i \in \{1, \ldots, N\}$, the aim of 3SMBC is to find the correct soft clustering solution for each instance in $\mathcal{X}^U$, which is defined later, according to either a known class or a new unknown cluster. The class[1] information of some instances is available in this kind of data. Therefore, $mathcalX$ can be divided into $\mathcal{X} = \mathcal{X}^L \cup \mathcal{X}^U$, where $\mathcal{X}^L = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(L)}\}$ is the subset of instances with an associated known class label and $\mathcal{X}^U = \{\mathbf{x}^{(L+1)}, \ldots, \mathbf{x}^{(N)}\}$ are the instances with unknown labels. This information is gathered in the mixture using a latent variable $\mathcal{Z}$. Therefore, this set can be divided into $\mathcal{Z} = \mathcal{Z}^L \cup \mathcal{Z}^U = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(L)}\} \cup \{\mathbf{z}^{(L+1)}, \ldots, \mathbf{z}^{(N)}\}$, separating known from unknown class labels, respectively, where $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_C^{(i)}, z_{C+1}^{(i)}, \ldots, z_K^{(i)})$, with $z_m^{(i)} = 1$ if instance i belongs to component m and with all other elements $z_{m'}^{(i)} = 0$, $\forall\ m' \neq m$. Note that $z_m^{(i)} = 0$ if $m > C$ for $\mathbf{z}^{(i)} \in \mathcal{Z}^L$, that is, $\forall i \in \{1, \ldots, L\}$, where C denotes the number of known classes in the semi-supervised setting. This constraint does not apply to $\mathcal{Z}^U$ because the instances with unknown class labels can belong to a known class or any other new group that can be discovered. For this reason, the final number of clusters is K, with $K \geq C$. The feature relevance for each mixture component is indicated in the set $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$, being $\mathbf{v}_m = (v_{m1}, \ldots, v_{mF})$ with $v_{mj} = 1$ if feature j is relevant to component m, and $v_{mj} = 0$ otherwise, $\forall\ m = 1, \ldots, K, j = 1, \ldots, F$. The values of each $\mathbf{v}_m$ are unknown, and, therefore, set $\mathcal{V}$ is a new set of latent variables. Besides, for each component and feature, $\rho_{mj} = p(v_{mj} = 1)$, the probability of feature j being relevant to component m, can be defined.

For the basic theory, there is no kind of subset selection or knowledge about the data to start with. For this reason, $\mathcal{V}$ does not exist, and $\mathcal{Z}^L$ is not a priori known. Therefore, the aim is to find $\mathcal{Z}$. In a finite mixture model, an instance is assumed to be generated by a probabilistic model given by a finite mixture of distributions. Assuming that the mixture has K components, the density function of an instance $\mathbf{x}^{(i)}$ is

$$p(\mathbf{x}^{(i)} \mid \Theta) = \sum_{m=1}^{K} \pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m), \tag{5.1}$$

where $p(\cdot)$ is the density function, $\boldsymbol{\theta}_m$ is the parameter set defining component m, and π_m is the mixing probability of component m, with $\pi_m \geq 0$ and $\sum_{m=1}^{K} \pi_m = 1$. The full parameter set of the mixture is $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \pi_1, \ldots, \pi_K\}$.

To explain Equation (5.1), π_m is the a priori probability that instance i was generated by component m. Then, $\mathbf{z}^{(i)}$, a binary random variable is defined as previously commented.

[1]Note that "cluster", "component", "group" and "class" are equivalent concepts at the end of the classification, but each concept will be used here to refer, respectively to a priori knowledge about instances (classes), mixture components (components) or identified groups (clusters).

Also, $p(z_m^{(i)} = 1) = \pi_m$. Therefore, it can be written

$$p(\mathbf{z}^{(i)}) = \prod_{m=1}^{K} \pi_m^{z_m^{(i)}}. \tag{5.2}$$

Similarly, the other term of Equation (5.1) can be written as $p(\mathbf{x}^{(i)} \mid z_m^{(i)} = 1) = p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m)$, which, extended, is

$$p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}, \Theta) = \prod_{m=1}^{K} p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m)^{z_m^{(i)}}. \tag{5.3}$$

Using Equations (5.2) and (5.3), Equation (5.1) can be rewritten by summing over all possible states of $\mathbf{z}^{(i)}$, as

$$\begin{aligned} p(\mathbf{x}^{(i)} \mid \Theta) &= \sum_{\mathbf{z}^{(i)}} p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} \mid \Theta) = \sum_{\mathbf{z}^{(i)}} p(\mathbf{z}^{(i)}) p(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)}, \Theta) \\ &= \sum_{\mathbf{z}^{(i)}} \left(\prod_{m=1}^{K} \pi_m^{z_m^{(i)}} \prod_{m=1}^{K} p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m)^{z_m^{(i)}} \right) \\ &= \sum_{m=1}^{K} \pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m). \end{aligned} \tag{5.4}$$

The whole set of variables $\mathbf{z}^{(i)}$ for all instances is defined as $\mathcal{Z} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\}$ and is the set of latent variables of the model.

The presented mixture of distributions has unknown parameters in Θ that must be estimated. This parameter set can be estimated using the maximum likelihood method. Therefore, assuming that each instance is independent and identically distributed (i.i.d.), and building the log-likelihood function ($\log L$) from Equation (5.4) and extending it to all the instances, $\log L$ is

$$\begin{aligned} \log L(\Theta \mid \mathcal{X}) &= \log p(\mathcal{X}|\Theta) \\ &= \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)} \mid \Theta) \\ &= \sum_{i=1}^{N} \log \left(\sum_{m=1}^{K} \pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m) \right). \end{aligned}$$

This log-likelihood function is difficult to maximize because the summation over the components is inside the logarithm function. This is because the equation deals with incomplete-data, due to the lack of knowledge about latent variables. For one of the first attempts to unify maximum likelihood estimation from incomplete-data, see [122]. Therefore, the log-likelihood function would change if both the latent variables ($\mathcal{Z}$) and the observable data

($\mathcal{X}$) were known, i.e., having the complete-data. Then, based on Equations (5.2) and (5.3), the complete-data log-likelihood can be defined as

$$\begin{aligned}\log L(\Theta \mid \mathcal{X}, \mathcal{Z}) &= \log p(\mathcal{X}, \mathcal{Z}|\Theta) \\ &= \log \prod_{i=1}^{N} \prod_{m=1}^{K} \pi_m^{z_m^{(i)}} p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m)^{z_m^{(i)}} \\ &= \sum_{i=1}^{N} \sum_{m=1}^{K} z_m^{(i)} \left(\log \pi_m + \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m) \right).\end{aligned} \tag{5.5}$$

The maximization of this complete-data log-likelihood function is straightforward because the summation is outside the logarithm. But since the latent variables are unknown, this function cannot be directly used. However, the expectation of this log-likelihood function can be obtained with respect to the posterior distribution of the latent variables. So, the EM algorithm, presented in Section 2.3.1, is used as an iterative algorithm to estimate the parameters that maximize the expectation of the complete-data log-likelihood function. This expectation is calculated in an iteration t, having fixed the parameters from the previous iteration[2] $t-1$, in the E-step of the EM algorithm. After this, the parameters of the distributions are recalculated to maximize this expectation (M-step). These two steps are repeated until a convergence criterion is reached.

Hence, the expectation of the complete-data log-likelihood function is given by

$$\begin{aligned}\mathcal{Q}(\Theta, \Theta^{t-1}) &= \mathbb{E}_{\mathcal{Z}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z})] \\ &= \sum_{\mathcal{Z}} p(\mathcal{Z} \mid \mathcal{X}, \Theta^{t-1}) \log p(\mathcal{X}, \mathcal{Z} \mid \Theta),\end{aligned} \tag{5.6}$$

where the posterior distribution of the latent variables given the data and the parameters of the previous iteration $t-1$ fixed, using Equation (5.5), is

$$p(\mathcal{Z} \mid \mathcal{X}, \Theta^{t-1}) \propto \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m) \right)^{z_m^{(i)}}. \tag{5.7}$$

This factorizes over i so that the $\{\mathbf{z}^{(i)}\}$ in this distribution are independent. Using this posterior distribution and using Bayes' theorem, the expected value of each $z_m^{(i)}$ can be calculated as

[2]Note that, for legibility, the notation related to iterations is used with Θ, but not with $\boldsymbol{\theta}$ throughout the document

$$\begin{aligned}
\mathbb{E}_{z_m^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\theta}_m}[z_m^{(i)}] &= \gamma(z_m^{(i)}) \\
&= \frac{\sum_{z_m^{(i)}} z_m^{(i)} (\pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m))^{z_m^{(i)}}}{\sum_{z_{m'}^{(i)}} (\pi_{m'} p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_{m'}))^{z_{m'}^{(i)}}} \\
&= \frac{\pi_m p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m)}{\sum_{m'=1}^{K} \pi_{m'} p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_{m'})} \\
&= p(z_m^{(i)} = 1 \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}_m),
\end{aligned}$$

which can be used to calculate the expectation of the complete-data log-likelihood as

$$\mathcal{Q}(\Theta, \Theta^{t-1}) = \sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_m^{(i)}) \left(\log \pi_m + \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_m) \right). \tag{5.8}$$

In the M-step the set of parameters is estimated to maximize the expectation of the complete-data log-likelihood, presented in Equation (5.8), as $\Theta^t = \arg\max_{\Theta} \mathcal{Q}(\Theta, \Theta^{t-1})$. The updated parameters are obtained by computing the partial derivatives of the expectation of the complete-data log-likelihood described above with respect to the different parameters and equaling to zero. These derivatives will be presented as part of the proposed solution for the specific problem.

5.3 Semi-supervised subspace soft clustering (3SMBC)

The above mixture model theory is applied to a clustering problem with two specific characteristics:

1. Groups of instances can be hidden in different feature subspaces. Therefore, a FSS is required in each mixture component. Hence, data structures that would remain hidden using all features or a single FSS might be identified.

2. The class information of some instances is available. This knowledge is used during the EM process to improve the final classification; therefore, this is a semi-supervised learning task.

Therefore, 3SMBC adds subspaces and available label instance information to model-based clustering (see Figure 5.1), plus a novel estimation of the final number of groups. The possibilities for adapting basic model-based clustering to the characteristics of the specific problem are described next.

The adaptation begins by including the search for subspaces in each component which leads on to the development for maximizing the above expectation of the complete-data log-likelihood function. Then, the available labeled instances information is taken into account to improve the final clustering. Besides, the estimation of the final number of components is provided, based on an iterative forward greedy approach.

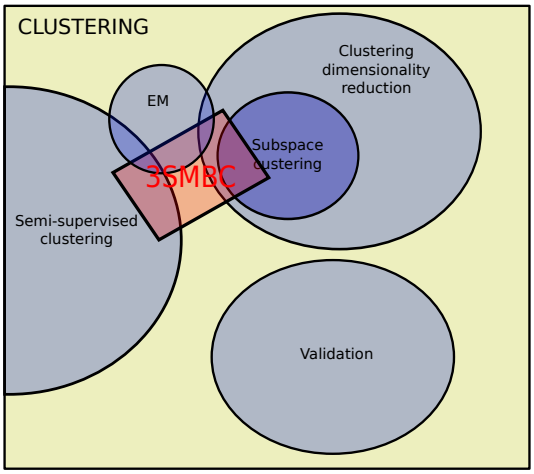


Figure 5.1: 3SMBC is a model-based clustering algorithm combining subspaces search and class labels information.

Adding subspaces to EM

To find the subspaces that best describe the components, the feature relevance for each each mixture component must be found. As previously mentioned, each component's feature relevance is indicated in the set $\mathcal{V}$, which is a new set of latent variables. Besides, for each component and feature, $\rho_{mj} = p(v_{mj} = 1)$, the probability that feature j is relevant to component m, can be defined. Then, assuming that features are independent given the component label, the next probability can be obtained for a component m and an instance i,

$$p(\mathbf{v}_m \mid z_m^{(i)} = 1) = \prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1 - \rho_{mj})^{1 - v_{mj}}.$$

This can be extended for all components as

$$p(\mathcal{V} \mid \mathbf{z}^{(i)}) = \prod_{m=1}^{K} \left(\prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1 - \rho_{mj})^{1 - v_{mj}} \right)^{z_m^{(i)}}. \tag{5.9}$$

Besides, Equation (5.3) can be extended by introducing $\mathcal{V}$, as

$$p(\mathbf{x}^{(i)} \mid \mathcal{V}, \mathbf{z}^{(i)}, \Theta) = \prod_{m=1}^{K} \left(\prod_{j=1}^{F} p(x_j^{(i)} \mid \theta_{mj})^{v_{mj}} p(x_j^{(i)} \mid \lambda_{mj})^{1 - v_{mj}} \right)^{z_m^{(i)}}, \tag{5.10}$$

where θ_{mj} indicates the density function parameters if attribute j is relevant to component m, whereas λ_{mj} indicates the density function parameters if attribute j is not relevant to component m. With the inclusion of subspaces, a whole new set of parameters have to be estimated: $\Theta = \{\theta_{mj}, \lambda_{mj}, \rho_{mj}, \pi_m\}_{m=1,\ldots,K; j=1,\ldots,F}$. The new density function, based on Equation (5.4), and using Equations (5.2), (5.9) and (5.10), is,

$$
\begin{aligned}
p(\mathbf{x}^{(i)} \mid \Theta) &= \sum_{\mathbf{z}^{(i)}} \sum_{\mathcal{V}} p(\mathbf{x}^{(i)}, \mathcal{V}, \mathbf{z}^{(i)} \mid \Theta) \\
&= \sum_{\mathbf{z}^{(i)}} \sum_{\mathcal{V}} p(\mathbf{x}^{(i)} \mid \mathcal{V}, \mathbf{z}^{(i)}, \Theta) p(\mathcal{V} \mid \mathbf{z}^{(i)}) p(\mathbf{z}^{(i)}).
\end{aligned}
$$

The summation over $\mathbf{z}^{(i)}$ is solved as in Equation (5.4), obtaining,

$$
\begin{aligned}
p(\mathbf{x}^{(i)} \mid \Theta) = \sum_{\mathcal{V}} \Bigg(\sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(& [\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})]^{v_{mj}} \\
& \times [(1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]^{1-v_{mj}} \Big) \Bigg).
\end{aligned}
$$

And the summation over $\mathcal{V}$ can be solved by summing over all the possible states of each v_{mj}, as,

$$
\begin{aligned}
p(\mathbf{x}^{(i)} \mid \Theta) &= \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \sum_{v_{mj}=0}^{1} \Big([\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})]^{v_{mj}} \\
&\quad \times [(1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]^{1-v_{mj}} \Big) \\
&= \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) + (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj}) \Big).
\end{aligned} \tag{5.11}
$$

Taking into account that each component can be described in a different feature subspace this is the new density function of an instance. The new log-likelihood function that should be maximized, by extending Equation (5.11) to all the instances, is

$$
\begin{aligned}
\log L(\Theta \mid \mathcal{X}) = \log p(\mathcal{X}|\Theta) &= \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)} \mid \Theta) \\
&= \sum_{i=1}^{N} \Bigg(\log \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) \\
&\quad + (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj}) \Big) \Bigg).
\end{aligned}
$$

This is again difficult to compute since the summation over the components is inside the logarithm function. This equation would change if the complete-data were known, i.e., knowing the sets of latent variables, $\mathcal{Z}$ and $\mathcal{V}$. Again by extending Equations (5.2), (5.9), and (5.10) to all the data,

$$
\begin{aligned}
p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) = & \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\prod_{j=1}^{F} p(x_j^{(i)} \mid \theta_{mj})^{v_{mj}} p(x_j^{(i)} \mid \lambda_{mj})^{1-v_{mj}} \right)^{z_m^{(i)}} \\
& \times \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1-\rho_{mj})^{1-v_{mj}} \right)^{z_m^{(i)}} \\
& \times \prod_{i=1}^{N} \prod_{m=1}^{K} \pi_m^{z_m^{(i)}},
\end{aligned}
$$

which can be simplified to

$$
\begin{aligned}
p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) = \prod_{i=1}^{N} \prod_{m=1}^{K} \Bigg(& \pi_m^{z_m^{(i)}} \prod_{j=1}^{F} \Big([\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})]^{v_{mj}} \\
& \times [(1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]^{1-v_{mj}} \Big)^{z_m^{(i)}} \Bigg).
\end{aligned}
\tag{5.12}
$$

The complete-data log-likelihood function can be obtained by taking the logarithm of previous function as,

$$
\begin{aligned}
\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V}) &= \log p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) \\
&= \log \prod_{i=1}^{N} \prod_{m=1}^{K} \Bigg(\pi_m^{z_m^{(i)}} \prod_{j=1}^{F} \Big([\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})]^{v_{mj}} \\
&\quad \times [(1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]^{1-v_{mj}} \Big)^{z_m^{(i)}} \Bigg),
\end{aligned}
$$

and operating again,

$$
\begin{aligned}
& \log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V}) \\
& \quad = \sum_{i=1}^{N} \sum_{m=1}^{K} \Big(z_m^{(i)} \log \pi_m \\
& \qquad + \sum_{j=1}^{F} \Big(z_m^{(i)} \Big[v_{mj} (\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
& \qquad + \ (1-v_{mj}) (\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})) \Big] \Big) \Big).
\end{aligned}
\tag{5.13}
$$

This indicates how necessary the two latent variables are: $z_m^{(i)}$ indicates instance i's mem-

bership of component m, whereas v_{mj} indicates attribute j's relevance for a component m. Obviously, the problem again is that these variables are not available and must be estimated. For this reason, and as in Equation (5.6), the expectation of the complete-data log-likelihood function must be calculated in the E-step of the EM algorithm, taking into account the two sets of latent variables $\mathcal{Z}$ and $\mathcal{V}$. This expectation is presented in the next section.

Expectation of the complete-data log-likelihood function

Similarly to Equation (5.6), the expectation can be written as

$$\mathcal{Q}(\Theta, \Theta^{t-1}) = \sum_{\mathcal{Z}} \sum_{\mathcal{V}} p(\mathcal{Z}, \mathcal{V} \mid \mathcal{X}, \Theta^{t-1}) \log p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta).$$

As in Equation (5.7), the posterior distribution of the latent variables given the data, having fixed the parameters of the previous iteration $t-1$, and using Equation (5.12), can be written as

$$p(\mathcal{Z}, \mathcal{V} \mid \mathcal{X}, \Theta^{t-1}) \propto \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\pi_m^{z_m^{(i)}} \prod_{j=1}^{F} \left([\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})]^{v_{mj}} \times [(1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]^{1-v_{mj}} \right)^{z_m^{(i)}} \right).$$

Before computing the expected values of each v_{mj} and each $z_m^{(i)}$, some other necessary probabilities must be defined:

$$p(x_j^{(i)}, v_{mj} = 1 \mid \theta_{mj}) = \rho_{mj} p(x_j^{(i)} \mid \theta_{mj}),$$

and, similarly

$$p(x_j^{(i)}, v_{mj} = 0 \mid \theta_{mj}) = (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj}).$$

Taking both expressions into account

$$\begin{aligned} p(x_j^{(i)} \mid \theta_{mj}) &= p(x_j^{(i)}, v_{mj} = 1 \mid \theta_{mj}) \\ &\quad + p(x_j^{(i)}, v_{mj} = 0 \mid \theta_{mj}) \\ &= \rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) + (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj}). \end{aligned}$$

Now, as detailed after Equation (5.7), the expected value of each v_{mj} can be calculated as

$$\begin{aligned}
&\mathbb{E}_{v_{mj},|\mathbf{x}_j^{(i)},\boldsymbol{\theta}_{mj}}[v_{mj}] = \gamma(v_{mj}) \\
&= \frac{\rho_{mj} p(x_j^{(i)} \mid \theta_{mj})}{\rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) + (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})} \\
&= p(v_{mj} = 1 \mid x_j^{(i)}, \theta_{mj}).
\end{aligned} \tag{5.14}$$

Using this, the expected value of each $z_m^{(i)}$ is

$$\begin{aligned}
&\mathbb{E}_{z_m^{(i)}|\mathbf{v}_m,\mathbf{x}^{(i)},\boldsymbol{\theta}_m}[z_m^{(i)}] = \gamma(z_m^{(i)}) \\
&= \frac{\pi_m \prod_{j=1}^{F} [\rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) + (1-\rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})]}{\sum_{m'=1}^{K} \pi_{m'} \prod_{j=1}^{F} [\rho_{m'j} p(x_j^{(i)} \mid \theta_{m'j}) + (1-\rho_{m'j}) p(x_j^{(i)} \mid \lambda_{m'j})]} \\
&= p(z_m^{(i)} = 1 \mid \mathbf{v}_m, \mathbf{x}^{(i)}, \boldsymbol{\theta}_m).
\end{aligned} \tag{5.15}$$

Thus the expectation of the complete-data log-likelihood, as in Equation (5.8) and using Equation (5.13), is

$$\begin{aligned}
\mathcal{Q}(\Theta, \Theta^{t-1}) = \sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_m^{(i)}) \\
\times \Bigg(\log \pi_m + \sum_{j=1}^{F} \Bigg(\gamma(v_{mj}) (\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
+ (1 - \gamma(v_{mj})) (\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})) \Bigg) \Bigg).
\end{aligned} \tag{5.16}$$

Then, for simplicity's sake,

$$\gamma(u_{mj}^{(i)}) = \gamma(z_m^{(i)}) \gamma(v_{mj}), \tag{5.17}$$

$$\gamma(w_{mj}^{(i)}) = \gamma(z_m^{(i)}) (1 - \gamma(v_{mj})). \tag{5.18}$$

Now Equation (5.16) can be simplified by introducing these new expressions and separating

each parameter into different addends

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X},\mathcal{Z},\mathcal{V})] \\
&= \sum_{i=1}^{N}\sum_{m=1}^{K} \gamma(z_m^{(i)}) \log \pi_m \\
&+ \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(u_{mj}^{(i)})(\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
&+ \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(w_{mj}^{(i)})(\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})).
\end{aligned} \tag{5.19}$$

Before detailing how to obtain the updated parameters of 3SMBC in the M-step, the expectation of the complete-data log-likelihood function is adapted to introduce the available instance label information.

Instance label information inclusion

3SMBC uses the available instance label information to guide the clustering of the unlabeled instances. Based on this information, the model learning process can be divided into two parts: the labeled instances are correctly classified into known classes $\{1,\ldots,C\}$ (classification term) and the unlabeled instances can be grouped either in those known or in other unknown components $\{1,\ldots,C,C+1,\ldots,K\}$ (clustering term). Thus, the expectation of the log-likelihood presented in Equation (5.19), separating the two learning steps, is,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X},\mathcal{Z},\mathcal{V})] \\
&\qquad = \mathbb{E}_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L,\mathcal{Z}^L,\mathcal{V})] \\
&\qquad\quad + \mathbb{E}_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U,\mathcal{Z}^U,\mathcal{V})],
\end{aligned}$$

where the expectation corresponding to the classification term is,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L,\mathcal{Z}^L,\mathcal{V})] \\
&= \sum_{i=1}^{L}\sum_{m=1}^{C} z_m^{(i)} \log \pi_m \\
&+ \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(u_{mj}^{(i)})(\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
&+ \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(w_{mj}^{(i)})(\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})).
\end{aligned} \tag{5.20}$$

Note that when $z_m^{(i)}$ is known, $\gamma(u_{mj}^{(i)})$ and $\gamma(w_{mj}^{(i)})$, calculated in Equations (5.17) and (5.18), are obtained using the value of $z_m^{(i)}$ instead of $\gamma(z_m^{(i)})$.

The expectation related to the clustering term is,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U, \mathcal{Z}^U, \mathcal{V})] \\
&= \sum_{i=L+1}^{N}\sum_{m=1}^{K} \gamma(z_m^{(i)}) \log \pi_m \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(u_{mj}^{(i)})(\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(w_{mj}^{(i)})(\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})).
\end{aligned} \tag{5.21}$$

Another feature of 3SMBC is the procedure for estimating the final number of clusters, which is described next.

Model order selection. Selection of the final number of clusters

The number of components of the finite mixture is an unknown parameter that must be estimated. This is a challenging task in clustering, which is very often tackled using a top-down approach, i.e., establishing a maximum number of components, K_{max}, and iteratively deleting one component in each step, depending on a quality measure, as the (regularized) log-likelihood function. In this proposal, and due to some available labeled instances, the minimum number of clusters is known beforehand. Therefore, from C, the minimum number of mixture components (one component per each known class), a bottom-up approach (see Algorithm 5.1) is proposed to estimate the final number of clusters using a greedy forward search. The process begins at an initial level $l = 0$, where a model $\mathcal{M}^0$ is built using a finite mixture with C components, each in different feature subspaces. Then, in $l = 1$, a new model $\mathcal{M}^1$ with $C + 1$ components is built. $\mathcal{M}^1$ tries to find a new component in a new feature subspace. This model is evaluated and compared with $\mathcal{M}^0$. If $\mathcal{M}^1$ is better than $\mathcal{M}^0$, then a new component is added to the group of known components, and level $l = 2$ starts with $C + 1$ known components trying to add another one. The convergence criterion is reached when $\mathcal{M}^l$ is better than $\mathcal{M}^{l+1}$, returning a clustering solution with $C + l$ components. Note that labeled instances can only belong to the C known components of level 0, whereas the unlabeled instances can belong to any component $C + l$, in level l.

Two models from consecutive levels and with a different number of components are compared by penalizing the log-likelihood with a term related to model complexity. BIC, presented in Section 2.3.1, is used for this purpose. BIC must be minimized, and being R the number of free parameters of the model, i.e., parameters that must be somehow estimated,

Algorithm 5.1 General code for model order selection. A model is represented by $\mathcal{M}^l$ where l is the level of the algorithm, K is the number of components of a model, C is the number of known classes at the beginning of the execution, BIC_{old} and BIC_{new} are the evaluation values of a model obtained with BIC.

$l = 0$
$K = C$
Build $\mathcal{M}^l$
$\text{BIC}_{\text{old}}, \text{BIC}_{\text{new}}$ = Evaluation of $\mathcal{M}^l$
repeat
  $\text{BIC}_{\text{old}} = \text{BIC}_{\text{new}}$
  $l = l + 1$
  $K = K + 1$
  Initialize K
  Build $\mathcal{M}^l$
  BIC_{new} = Evaluation of $\mathcal{M}^l$
until $\text{BIC}_{\text{old}} \geq \text{BIC}_{\text{new}}$

and N the number of instances, its definition is

$$BIC = -2 \log L + R \log N. \tag{5.22}$$

At level 0, parameters of the C known components must be initialized. This is carried out using only labeled instances depending on the given labels. On the other hand, new component initialization of each $\mathcal{M}^l$, with $l \geq 1$, must be explained in detail, because it is performed using those instances that worse fit the components of level $l-1$. Therefore, for a level l, it is assumed that instances that worse fit to the $l-1$ components are candidates for belonging to a new component. Based on this assumption the unlabeled instances are ranked, taking into account the sum of all membership values to the $l-1$ components. This is calculated, from Equation (5.11) and for an instance i, as

$$\sum_{m=1}^{C+(l-1)} \left(\pi_m \prod_{j=1}^{F} [\rho_{mj} p(x_j^{(i)} \mid \theta_{mj}) + (1 - \rho_{mj}) p(x_j^{(i)} \mid \lambda_{mj})] \right). \tag{5.23}$$

The top-ranked instances are the ones that worse fit the known components. Then, beginning from the top of the ranking, some of instances are chosen to initialize the subspace for the new component. The number of instances is the candidates threshold (CTh).

Finally, to generalize for a level l of 3SMBC, Equations (5.20) and (5.21), for $\mathcal{M}^l$, are

updated as

$$\begin{aligned}
&\mathbb{E}^{l}_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L, \mathcal{Z}^L, \mathcal{V})] \\
&= \sum_{i=1}^{L}\sum_{m=1}^{C} z_m^{(i)} \log \pi_m \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(u_{mj}^{(i)})(\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(w_{mj}^{(i)})(\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})),
\end{aligned} \tag{5.24}$$

$$\begin{aligned}
&\mathbb{E}^{l}_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U, \mathcal{Z}^U, \mathcal{V})] \\
&= \sum_{i=L+1}^{N}\sum_{m=1}^{C+l} \gamma(z_m^{(i)}) \log \pi_m \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{C+l}\sum_{j=1}^{F} \gamma(u_{mj}^{(i)})(\log \rho_{mj} + \log p(x_j^{(i)} \mid \theta_{mj})) \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{C+l}\sum_{j=1}^{F} \gamma(w_{mj}^{(i)})(\log(1-\rho_{mj}) + \log p(x_j^{(i)} \mid \lambda_{mj})).
\end{aligned} \tag{5.25}$$

Now E and M steps for the adapted EM algorithm of 3SMBC are detailed next. For simplicity's sake, it is assumed that $l = 1$ in the following.

E-step

The aim in E-step of EM algorithm is to calculate the expectation of the above complete-data log-likelihood function, having fixed the parameters. These parameters must be initialized before the first iteration of the EM algorithm, whereas they are recalculated in M-step for subsequent iterations.

The value of the expected complete-data log-likelihood function for $\mathcal{M}^1$ is the sum of Equations (5.24) and (5.25). Latent variables expectations are in Equations (5.15), (5.17), and (5.18).

M-step

Parameters are recalculated in M-step to maximize the value of the expectation of the complete-data log-likelihood function. As already mentioned, these updates are obtained by computing the partial derivatives of this expectation and equaling to zero. The univariate Gaussian distribution for each feature and component is used for this explanation. Therefore $\theta_{mj} = (\mu_{\theta_{mj}}, \sigma^2_{\theta_{mj}})$, and

$$\log p(x_j^{(i)} \mid \theta_{mj}) = \log(\sigma_{\theta_{mj}}^{-1}(2\pi)^{-\frac{1}{2}}) - \frac{1}{2}(x_j^{(i)} - \mu_{\theta_{mj}})^2\sigma_{\theta_{mj}}^{-2},$$

- π_m is updated[3] using a Lagrange multiplier to enforce constraint $\sum_{m=1}^{C+1}\pi_m = 1$:

$$\frac{\partial}{\partial \pi_m}\left(\sum_{i=1}^{L}\sum_{m=1}^{C} z_m^{(i)}\log\pi_m + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\gamma(z_m^{(i)})\log\pi_m + \lambda\left(\sum_{m=1}^{C+1}\pi_m - 1\right)\right) = 0, \quad \forall m = 1,\ldots,C+1,$$

whose derivative is

$$\sum_{i=1}^{L} z_m^{(i)}\frac{1}{\pi_m} + \sum_{i=L+1}^{N}\gamma(z_m^{(i)})\frac{1}{\pi_m} + \lambda = 0.$$

Multiplying both sides by π_m and summing over m, with $m = 1,\ldots,C+1$, then $\lambda = -N$, as

$$-\lambda = \sum_{i=1}^{L}\sum_{m=1}^{C+1} z_m^{(i)} + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\gamma(z_m^{(i)}) = N,$$

and then each π_m is updated using

$$\pi_m = \frac{\sum_{i=1}^{L} z_m^{(i)} + \sum_{i=L+1}^{N}\gamma(z_m^{(i)})}{N}. \tag{5.26}$$

- $\mu_{\theta_{mj}}$ is updated solving the following partial derivative equation:

$$\frac{\partial}{\partial \mu_{\theta_{mj}}}\left(\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(u_{mj}^{(i)})\log p(x_j^{(i)} \mid \theta_{mj}) + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(u_{mj}^{(i)})\log p(x_j^{(i)} \mid \theta_{mj})\right) = 0.$$

Then the result is

[3]Note that the classification term only iterates theoretically until $m = C$, but it can be assumed that this iteration finishes at $m = C+1$ with $z_{C+1}^{(i)} = 0$, $\forall i = 1,\ldots,L$.

$$\sum_{i=1}^{L}\Big(\gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}x_j^{(i)} - \gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}\mu_{\theta_{mj}}\Big) + \sum_{i=L+1}^{N}\Big(\gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}x_j^{(i)} - \gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}\mu_{\theta_{mj}}\Big) = 0,$$

and the value of the parameter can be found as,

$$\begin{aligned}\mu_{\theta_{mj}} &= \frac{\sum_{i=1}^{L}\gamma(u_{mj}^{(i)})x_j^{(i)} + \sum_{i=L+1}^{N}\gamma(u_{mj}^{(i)})x_j^{(i)}}{\sum_{i=1}^{L}\gamma(u_{mj}^{(i)}) + \sum_{i=L+1}^{N}\gamma(u_{mj}^{(i)})} \\ &= \frac{\sum_{i=1}^{N}\gamma(u_{mj}^{(i)})x_j^{(i)}}{\sum_{i=1}^{N}\gamma(u_{mj}^{(i)})}, \quad \forall m = 1,\ldots,C+1; j = 1,\ldots,F.\end{aligned} \tag{5.27}$$

- And for $\sigma_{\theta_{mj}}^2$,

$$\frac{\partial}{\partial\sigma_{\theta_{mj}}^2}\Bigg(\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(u_{mj}^{(i)})\log p(x_j^{(i)} \mid \theta_{mj}) + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(u_{mj}^{(i)})\log p(x_j^{(i)} \mid \theta_{mj})\Bigg) = 0.$$

The derivative is

$$\sum_{i=1}^{L}\Big(\gamma(u_{mj}^{(i)})(x_j^{(i)} - \mu_{mj})^2\sigma_{\theta_{mj}}^{-4} - \gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}\Big) + \sum_{i=L+1}^{N}\Big(\gamma(u_{mj}^{(i)})(x_j^{(i)} - \mu_{mj})^2\sigma_{\theta_{mj}}^{-4} - \gamma(u_{mj}^{(i)})\sigma_{\theta_{mj}}^{-2}\Big) = 0,$$

and the parameter update is,

$$\begin{aligned}\sigma_{\theta_{mj}}^2 &= \frac{\sum_{i=1}^{L}\gamma(u_{mj}^{(i)})(x_j^{(i)} - \mu_{\theta_{mj}})^2 + \sum_{i=L+1}^{N}\gamma(u_{mj}^{(i)})(x_j^{(i)} - \mu_{\theta_{mj}})^2}{\sum_{i=1}^{L}\gamma(u_{mj}^{(i)}) + \sum_{i=L+1}^{N}\gamma(u_{mj}^{(i)})} \\ &= \frac{\sum_{i=1}^{N}\gamma(u_{imj})(x_j^{(i)} - \mu_{\theta_{mj}})^2}{\sum_{i=1}^{N}\gamma(u_{mj}^{(i)})}, \quad \forall m = 1,\ldots,C+1; j = 1,\ldots,F.\end{aligned} \tag{5.28}$$

The same development is valid for $\lambda_{mj} = (\mu_{\lambda_{mj}}, \sigma^2_{\lambda_{mj}})$ but using $\gamma(w^{(i)}_{mj})$ instead of $\gamma(u^{(i)}_{mj})$ to indicate that feature j is irrelevant for component m.

- Then, $\mu_{\lambda_{mj}}$ is updated as

$$\mu_{\lambda_{mj}} = \frac{\sum_{i=1}^{N} \gamma(w^{(i)}_{mj}) x^{(i)}_j}{\sum_{i=1}^{N} \gamma(w^{(i)}_{mj})}, \quad \forall m = 1, \ldots, C+1; j = 1, \ldots, F. \tag{5.29}$$

- And $\sigma^2_{\lambda_{mj}}$

$$\sigma^2_{\lambda_{mj}} = \frac{\sum_{i=1}^{N} \gamma(w^{(i)}_{mj})(x^{(i)}_j - \mu_{\lambda_{mj}})^2}{\sum_{i=1}^{N} \gamma(w^{(i)}_{mj})}, \quad \forall m = 1, \ldots, C+1; j = 1, \ldots, F. \tag{5.30}$$

- Finally in $\mathcal{M}^1$, ρ_{mj} is updated by

$$\begin{aligned}
\frac{\partial}{\partial \rho_{mj}} \Bigg(& \sum_{i=1}^{L} \sum_{m=1}^{C} \sum_{j=1}^{F} \gamma(u^{(i)}_{mj}) \log \rho_{mj} \\
& + \sum_{i=L+1}^{N} \sum_{m=1}^{C+1} \sum_{j=1}^{F} \gamma(u^{(i)}_{mj}) \log \rho_{mj} \\
& + \sum_{i=1}^{L} \sum_{m=1}^{C} \sum_{j=1}^{F} \gamma(w^{(i)}_{mj}) \log(1 - \rho_{mj}) \\
& + \sum_{i=L+1}^{N} \sum_{m=1}^{C+1} \sum_{j=1}^{F} \gamma(w^{(i)}_{mj}) \log(1 - \rho_{mj}) \Bigg) = 0,
\end{aligned}$$

whose partial derivative solution is,

$$\sum_{i=1}^{N} \gamma(u^{(i)}_{mj}) \frac{1}{\rho_{mj}} - \sum_{i=1}^{N} \gamma(w^{(i)}_{mj}) \frac{1}{1 - \rho_{mj}} = 0.$$

This parameter is updated by

$$\rho_{mj} = \frac{\sum_{i=1}^{N} \gamma(u^{(i)}_{mj})}{\sum_{i=1}^{L} z^{(i)}_m + \sum_{i=L+1}^{N} \gamma(z^{(i)}_m)}, \quad \forall m = 1, \ldots, C+1; j = 1, \ldots, F. \tag{5.31}$$

Note that $z^{(i)}_{C+1} = 0$ for $i = 1, \ldots, L$ for the three sets of parameters, θ_{mj}, λ_{mj} and ρ_{mj}.

5.4 Experimental results

5.4.1 Data

Both synthetic and real data sets are used to evaluate the proposed algorithm. All these data are fully labeled, and the *cluster assumption* (see Section 2.4.1) is again assumed. Thus, original labels will be used as ground truth to evaluate the clustering solutions. As the input of 3SMBC is expected to be a partially labeled data set, some of the labels will be hidden to the algorithm in each experiment.

The synthetic data data have been randomly generated using the Weka [262] data generator tool. Synthetic data sets were created by using different features, creating subspaces. The number of features depends on the experiment and details are indicated in each subsection. All relevant features were randomly generated using the Gaussian distribution with random mean and standard deviation, between 1 and 10, and 0.2 and 0.5, respectively. The irrelevant features were randomly generated using the Gaussian distribution with mean 0 and standard deviation 1, unless otherwise stated.

Real data sets are collected from the UCI repository [91] (see Table 5.1). The selected data sets are very typical examples used for evaluating different pattern recognition tasks. Although these data cannot be considered high-dimensional data, it is also interesting to check the behaviour of the proposal with this kind of data whose classes are not balanced and subspaces are not a main characteristic for them. However, in trying to introduce some complexity into the task, 25 Gaussian-distributed features were added to the original number of features for each data set, simulating some noise for the experiment. Therefore, the used data are called *diabetes25*, *iris25*, and *shape25*.

Data	N	F	K
diabetes	768	8	2
iris	150	4	3
shape	160	17	9

Table 5.1: Number of instances (N), features (F), and classes (K) of real data sets used in the experiments.

5.4.2 Evaluation process

Experiments are divided into two sets. The first of them is based on evaluating the proposed algorithm under different data conditions, like different data distributions, clusters in very low dimensional spaces, or clusters with different number of dimensions. Therefore, different scenarios are simulated to assess the 3SMBC behaviour under different data conditions (Section 5.4.3).

It is also interesting to compare the behaviour of 3SMBC against a related algorithm. Thus, the second set of experiments consists of a comparison of 3SMBC and CLWC (see Section 2.5.1). CLWC has been chosen due to the similarity of the algorithm regarding the weighting scheme for features and the available information exploitation (using pairwise constraints in this case). Both synthetic and real data sets were used in this comparison and details can be found in Section 5.4.4.

All the used data are fully labeled. All these original labels will be used as ground truth to evaluate the output clusters. However, as the input of 3SMBC is expected to be a partially labeled data set, some of the labels will be hidden to the algorithm in each experiment. Instances that retain their original labels are randomly selected and, on this ground, each scenario is executed 10 times (for 10 random selections), unless otherwise stated, to account for variability.

After some preliminary studies, CTh = 5 (see Section 5.3) has been chosen for these experiments. This value can be considered as a trade-off between information and noise for the initialization of a new component.

3SMBC is based on soft clustering. For the purpose of comparison with other algorithms, algorithm outputs were then translated, using the group with highest a posteriori probability in each case, to assign only one cluster to each instance (hard clustering). ARI, detailed in Section 2.3.2, was used to compare the post-processed output with the original labels.

5.4.3 Results

Synthetic scenarios under different data conditions

Different scenarios have been created to simulate several data conditions for an interesting evaluation of the 3SMBC performance. These scenarios cover many situations that can arise when using our algorithm, results show the 3SMBC reliability with the used data in the created scenarios.

Different data distributions

This experiment aims at evaluating the behaviour of 3SMBC when irrelevant features are generated by using different distributions. Specifically, Gaussian and Uniform distributions were considered to generate irrelevant data. According to this, four different data sets, each with 400 instances and four balanced classes, were generated: *uni25* and *gauss25* have 25 features, with 40% of relevant features per class and 20% of all features are completely irrelevant. On the other hand, *uni50* and *gauss50* have 50 features, with 20% of relevant features per class and 60% of completely irrelevant features. All irrelevant features were generated using the Uniform (between 0 and 1) and the Gaussian distribution (with mean 0 and standard deviation 1) depending on each data set and as indicated in each name. The relevant features were generated as indicated in Section 5.4.1.

Labels of one class are completely hidden for all cases, and two different percentages of labeled instances are used, 20% and 40%. The results are shown in Table 5.2.

The main conclusion is that there was no significant difference in the results obtained when irrelevant features were generated using Uniform and Gaussian distributions, regardless of how the model of 3SMBC is built using the Gaussian distribution for all features. Another conclusion is that results for data sets with 50 features were better than results for data sets with 25 features. This conclusion should be taken wisely since these results may depend on the instances selected to maintain their labels. A scalable experiment is shown next to study

	Data			
%	*uni25*	*gauss25*	*uni50*	*gauss50*
20	0.90 ± 0.04	0.93 ± 0.03	0.99 ± 0.02	0.99 ± 0.02
40	0.94 ± 0.04	0.92 ± 0.00	0.97 ± 0.03	1.00 ± 0.00

Table 5.2: Average ± standard deviation of ARI results when clustering data with 3SMBC using different distributions (25 or 50 features and Uniform or Gaussian models for irrelevant features) and percentage of labeled instances.

the behaviour of 3SMBC when the number of features change. The difference between using 20% and 40% of labeled instances was negligible for all scenarios but for *uni25*, where the ARI obtained with 20% and 40% of labeled instances were 0.90 ± 0.04 and 0.94 ± 0.04, respectively. The small ARI differences between results with different percentage of labeled instances indicates the correct behaviour of 3SMBC even when the available information is not very high.

Scalability

Although the purpose of this work is not to design a scalable algorithm, an experiment was carried out to check the behaviour of the algorithm when data sets with different number of instances and features are clustered. Nine different synthetic data sets were generated, combining 100, 500, and 1000 instances, with 10, 50, and 100 features. Data were generated in four balanced classes. The percentage of relevant features for each data set was between 60% and 80%. All experiments were run with 40% of labeled instances and one class was completely hidden to the algorithm.

	F		
N	10	50	100
100	1.00 ± 0.00	0.96 ± 0.04	0.94 ± 0.02
500	0.94 ± 0.04	0.95 ± 0.05	0.95 ± 0.05
1000	0.93 ± 0.04	0.94 ± 0.05	0.94 ± 0.01

Table 5.3: Average ± standard deviation of ARI results when clustering data with 3SMBC for different number of instances (N) and features (F).

The results in Table 5.3 show the good behaviour of the algorithm regardless of the number of instances and features. The basic case, with 100 instances and 10 features, was clustered obtaining a perfect matching with the original labels. Results slightly decreased from this basic case when the number of instances was increased. Thus, with 10 features, the ARI was 0.94 and 0.93 on average with 500 and 1000 instances, respectively, both with a standard deviation of 0.04. The decrease was similar when the number of instances was maintained but the number of features increased. Thus, with 100 instances, the ARI was

0.96 ± 0.04 and 0.94 ± 0.02 for data with 50 and 100 features, respectively. There were not significant differences when data combined 500 and 1000 instances with 50 and 100 features, obtaining results around 0.95 on average with 500 instances and 0.94 with 1000 instances.

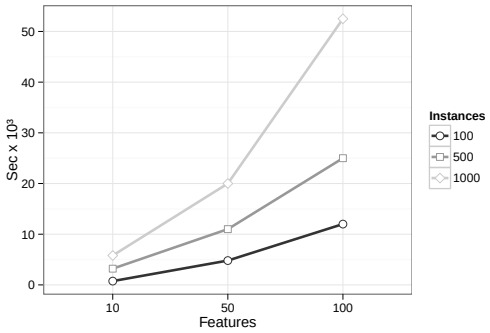


Figure 5.2: Computational cost in time per level of 3SMBC depending on the number of instances and features.

3SMBC is a computationally demanding algorithm. This cost is known for EM-based algorithms, and is increased here due to the iterative process to estimate the relevance of each feature and the final number of clusters. This cost is assumed, and the current version of 3SMBC did not try to overcome this situation that can be considered as future work. The computational cost for completing one level of 3SMBC is detailed in Figure 5.2. This was calculated according to the experimentation run on a multi-core machine with eight Intel(R) Xeon(R) CPU E5320 @1.86GHz and 12GB of RAM. Values were approximated since the time cost for a level with fewer components is lower than for another level with more components. According to these approximated results, the time cost is higher when the number of features is increased than with more instances. Nevertheless, note the differences on execution time depending on both the number of instances and features and how they are combined.

Different number of dimensions

Now the 3SMBC performance is assessed when the number of relevant features for each class differs. For this purpose, a synthetic data set was generated with 120 instances in three balanced classes, which are characterized by 30 features with the next characteristics: class 1 has 10% of relevant features (3 features), class 2 has 40% of relevant features (12 features), and class 3 has 85% of relevant features (25 features). In total, 95% of features are relevant for some class (28 features).

The validation scenario was created by running 3SMBC 15 times, with 40% of labeled instances and completely hiding one class for each execution (each class was completely hidden five times).

Class 1 (10%)	Class 2 (40%)	Class 3 (85%)	Total
0.98 ± 0.02	0.97 ± 0.03	0.96 ± 0.04	0.97 ± 0.03

Table 5.4: Average $\pm$ standard deviation of ARI results when clustering data with 3SMBC for clusters with different number of relevant features. The percentage of relevant features of the class that is completely hidden to the algorithm is shown within parentheses.

Results are shown in Table 5.4, where the ARI results are presented depending on the class completely hidden to the algorithm. There were some differences in the results, since

the performance slightly decreased when the hidden class was characterized with a higher number of features. Furthermore, if total average results are calculated, and according to this experiment, 3SMBC is able to correctly cluster data when clusters are characterized with different number of dimensions and one class must be discovered, regardless of which class is completely hidden.

Very low dimensional space

This experiment aims at evaluating the behaviour of the proposal when clusters are characterized by a very low number of features compared with the total number of available features. Three data sets were generated with 60 instances, three balanced classes, and 20, 200, and 2000 features respectively. Each cluster was characterized with 10%, 1%, and 0.1% of the total number of features, respectively, for each data set (i.e., 2 features). The cases with only 1%, and 0.1% of relevant features for each class are extreme scenarios since finding the correct features in such a big search space is a very tough task.

10%	1%	0.1%
1.00 ± 0.00	0.40 ± 0.04	0.25 ± 0.07

Table 5.5: Average ± standard deviation of ARI results when clustering data with 3SMBC for clusters in very low relevant dimensional spaces.

Results are shown in Table 5.5. 3SMBC obtained perfect clustering solutions when clusters were characterized by 10% of features. However, the ARI values sharply dropped when only 1% and 0.1% of features were relevant. As previously commented, these are difficult scenarios that are open to further studies related to 3SMBC and other algorithms to find better solutions.

5.4.4 3SMBC vs CLWC

The algorithm CLWC, presented in Section 2.5.1, is used for comparison purpose. This algorithm is very related to 3SMBC because subspaces are found by weighting features to indicate the relevance to each cluster. Besides, pairwise constraints are used as information known beforehand to guide the clustering process. These constraints can be directly obtained from partially labeled data sets. All pairs of instances with the same label have a must-link constraint, whereas all pairs of instances with different labels have a cannot-link constraint. These labels and constraints are mapped in data sets used to evaluate 3SMBC. Thus, exactly the same data sets were used to evaluate 3SMBC and CLWC.

Another characteristic of CLWC is that it needs to fix the number of clusters beforehand. Although this is a crucial parameter that does not need to be fixed in 3SMBC, it is known when original data are completely labeled. Then two different results for CLWC are shown depending on the number of clusters. The first result is obtained by calculating the average and standard deviation of 50 executions obtained by running CLWC with the correct number of clusters (K) as input parameter. Note that CLWC must be run several times because different cluster initializations are used. However, this comparison is not completely fair,

since 3SMBC automatically estimates the number of clusters and it is a fixed parameter for 3SMBC. For this reason, a second result is also presented for CLWC that is obtained by calculating the average and standard deviation of 150 executions: 50 executions obtained with $K-1$, 50 executions obtained with K, and, finally, 50 executions obtained with $K+1$.

Experiments are divided depending on the data, synthetic and real, used for comparison.

Synthetic data comparison

The synthetic data set generated for the comparison has 125 instances divided into five balanced classes. The number of features for each cluster is 25, with 20% of irrelevant features and 40% of relevant features.

This experiment aims at comparing the behaviour of 3SMBC and CLWC when different percentages of labeled instances (20% and 40%) are available, and when different number of classes must be completely discovered, i.e., there are not any labeled instances for these classes (different scenarios are created by completely hiding from one to three classes for each percentage of labeled instances).

Results are shown in Table 5.6, where CLWC$_K$ and CLWC$_{\{K,K\pm1\}}$ refer to results obtained with CLWC fixing the correct number of clusters (K) and fixing $\{K-1, K, K+1\}$, respectively. Regarding 3SMBC results, it can be seen how the algorithm was able to obtain an acceptable clustering solution (0.90 ± 0.12) even when there were three classes completely unknown and only a 20% of labeled instances. The ARI value was 0.96 ± 0.02 when there was only one unknown class and 40% of labeled instances. The behaviour of CLWC was very dependent on the initialization. This is evident from the high standard deviations obtained in all scenarios, meaning that, on some occasions, CLWC was able to obtain a very accurate clustering solution while other solutions obtained a very low performance for the same scenario. On average, CLWC results were worse than those obtained by 3SMBC even when the correct number of clusters was fixed beforehand. As expected, when CLWC worked with $\{K-1, K, K+1\}$ clusters, the difference with 3SMBC was even more sizeable. A curious behaviour of CLWC was that when three classes were hidden to the input data ($H = 3$)), the obtained ARI values were higher with 20% of labeled instances (mean ARI of 0.68) than with 40% (mean ARI of 0.53).

	20%			40%		
Algorithm	H=1	H=2	H=3	H=1	H=2	H=3
3SMBC	0.93 ± 0.04	0.92 ± 0.01	0.90 ± 0.12	0.96 ± 0.02	0.95 ± 0.04	0.91 ± 0.10
CLWC$_K$	0.82 ± 0.18	0.71 ± 0.30	0.74 ± 0.24	0.86 ± 0.15	0.77 ± 0.25	0.59 ± 0.41
CLWC$_{\{K,K\pm1\}}$	0.75 ± 0.25	0.71 ± 0.27	0.68 ± 0.28	0.77 ± 0.25	0.70 ± 0.28	0.53 ± 0.40

Table 5.6: ARI values of 3SMBC and CLWC for synthetic data depending on the percentage of labeled instances (20% and 40%) and the number of classes with no representation in the labeled data (H).

Together with ARI results, it is interesting to show the behaviour of 3SMBC regarding

	20%			40%		
Algorithm	H=1	H=2	H=3	H=1	H=2	H=3
3SMBC	3.00 ± 0.86	4.33 ± 0.58	2.66 ± 0.57	1.66 ± 0.5	3.22 ± 1.48	3.33 ± 1.00

Table 5.7: Number of new clusters estimated by 3SMBC in synthetic data depending on the percentage of labeled instances (20% and 40%) and the number of classes with no representation in the labeled data (H).

the estimation of the number of clusters. Table 5.7 shows the number of new clusters, i.e., clusters that were not represented in the labeled data, estimated by 3SMBC for the results presented in Table 5.6. 3SMBC overestimated the number of clusters for all cases except when 20% of instances were labeled with $H = 3$. This is the worst scenario for 3SMBC with a considerable number of clusters to be identified and only a little of known information.

Although finding the exact number of clusters could be an issue to obtain good clustering solutions, note that it is better to overestimate the final number of clusters than to underestimate it. This is related to the *cluster assumption*, which states that data with high similarity, i.e., data within the same cluster, must share the same class label. This assumption means that instances with different class labels shall not be grouped into the same cluster. However, this assumption does not imply that every single class forms a single cluster.

Real data comparison

The created scenarios regarding the labeled instances are the same as before for the synthetic data, with two percentages of labeled instances (20% and 40%). Two cases are also compared regarding the number of classes hidden to the input data (0 and 1).

Results are shown in Table 5.8. As expected, for a similar scenario regarding the percentage of labeled instances and hidden classes, ARI results completely depended on the data set. However, similar conclusions about 3SMBC and the comparison with CLWC can be drawn from all data. 3SMBC outperformed on average $\text{CLWC}_{\{K,K\pm1\}}$ in all scenarios and data except for *shape25* with 20% of labeled instances. Besides, 3SMBC also obtained very competitive results when compared with CLWC_K, although the comparison was not completely fair since the correct number of clusters was fixed beforehand for CLWC. This comparison became even more important with $H = 1$, since a cluster was not represented in the known information and, in theory, the clustering problem became more difficult. 3SMBC outperformed on average CLWC_K in all data with $H = 1$ and 40% of labeled instances, even when CLWC_K had the correct information about the number of clusters as input. It indicates that the initialization step in 3SMBC for the new and unknown cluster correctly worked. Finally, another interesting result is, as for synthetic data, related to the standard deviation. Due to differences in the initialization, CLWC outputs a higher standard deviation in general than 3SMBC, i.e., the behaviour of 3SMBC can be considered more stable. This is a desirable state in clustering since, for real applications, without the full set of correct

labels, the validation for the number of clusters and initialization is not a trivial task.

		20%		40%	
Data	Algorithm	H=0	H=1	H=0	H=1
diabetes25	3SMBC	0.19 ± 0.08	0.14 ± 0.02	0.34 ± 0.06	0.36 ± 0.01
	CLWC_K	0.21 ± 0.03	0.05 ± 0.02	0.41 ± 0.01	0.32 ± 0.10
	$\text{CLWC}_{\{K,K\pm1\}}$	0.12 ± 0.09	0.07 ± 0.08	0.24 ± 0.18	0.22 ± 0.17
iris25	3SMBC	0.84 ± 0.06	0.83 ± 0.05	0.91 ± 0.05	0.89 ± 0.05
	CLWC_K	0.89 ± 0.14	0.85 ± 0.21	0.90 ± 0.17	0.84 ± 0.22
	$\text{CLWC}_{\{K,K\pm1\}}$	0.72 ± 0.24	0.74 ± 0.19	0.64 ± 0.35	0.72 ± 0.23
shape25	3SMBC	0.52 ± 0.01	0.61 ± 0.03	0.75 ± 0.02	0.72 ± 0.02
	CLWC_K	0.64 ± 0.14	0.69 ± 0.09	0.70 ± 0.10	0.62 ± 0.25
	$\text{CLWC}_{\{K,K\pm1\}}$	0.63 ± 0.15	0.67 ± 0.12	0.68 ± 0.14	0.66 ± 0.22

Table 5.8: ARI values of 3SMBC and CLWC for real data depending on the percentage of labeled instances (20% and 40%) and the number of classes with no representation in the labeled data (H).

5.5 Summary and discussion

A semi-supervised clustering algorithm, called 3SMBC, capable of discovering new classes, based on EM theory, and including searching for subspaces, has been presented in this chapter. Besides, 3SMBC is included within a process in which the number of final clusters is automatically selected depending on the BIC criterion. This process is not only important because the number of clusters is automatically selected, but also because the new found clusters are initialized based on those instances that worse fit previous models. This initialization leads to find data structures that were not represented in the initial available information.

Some conclusions can be repeated from those obtained with KMF. The reliability of the available labels is even more important for 3SMBC since, in the current approach, labeled instances cannot change their labels during the algorithm. Although this fact, together with the initialization, also guide the search for subspaces, this process is integrated within 3SMBC instead of using separately the labels as in KMF.

the accuracy also decreases. Although this is also a normal situation due to the difficulty of searching when the search space grows, the decrease is not as biased to the number of features as to the number of new clusters that must be found. In any case, and as can be seen in Section 5.4.3, the estimation of the final number of clusters was very accurate obtaining a perfect estimation in many scenarios, even when there were clusters that were not represented in the available data labels. This good estimation also leads to very accurate results, obtaining again perfect clustering solutions when compared to the original labels.

Although 3SMBC is based on soft clustering, results were post-processed to hard clustering in order to validate and compare results with other algorithms. However, there was no lost of accuracy during this translation since all obtained results were almost like in hard

clustering, i.e., the highest value of the probability of membership of an instance was virtually 1 for a specific component, while the other components obtained almost a 0 in consequence.

Several evaluations (using ARI) of our method were provided under different data conditions creating some useful scenarios to understand the general behaviour of the algorithm. The first experiment was based on testing 3SMBC by clustering data generated using different distributions. Although the Gaussian distribution is assumed in the theory of 3SMBC, the obtained results when clustering data generated by other distribution were, at least, as competitive as results when clustering data generated with the Gaussian distribution.

The scalability was taken into account for the next experiment. Different synthetic data sets were generated starting with 100 instances and 10 features, and increasing both parameters until 1000 instances and 100 features. Although there were slight decreases of ARI values when the parameters were increased, the validation results can be considered as high. However, the time cost was very demanding for those cases in which the number of instances and features grew. 3SMBC was not created for being a fast algorithm, and the demanding computational cost is assumed.

Another scenario was created trying to simulate a data set containing three clusters with different density regarding features, i.e., clusters were generated with very different number of features: 10%, 40%, and 85%, respectively, with respect to the total number of features. Each cluster was hidden to 3SMBC in the input data to check whether the algorithm was able to correctly cluster data, even when a cluster with different number of features was completely hidden. Results showed that 3SMBC obtained accurate results regardless of the hidden cluster.

Data in a very low dimensional space was another experiment to check the behaviour of 3SMBC. Very extreme scenarios were created by using only 10%, 1%, and 0.1% relevant features, with respect to the total number of features, to generate each cluster. Although results yielded a perfect solution for the case with clusters generated with 10% of relevant features, the validation results with 1% and 0.1% of relevant features were not so accurate demonstrating the difficulty of these scenarios, where clusters are very hidden within the total space of features.

Another algorithm, called CLWC, was also used for comparison purposes. Both synthetic and real data sets were used for the comparison. Again, different scenarios were created depending on the percentage of labeled instances and the number of classes completely hidden to the algorithms in the input data. Although the correct number of clusters was fixed beforehand to CLWC, 3SMBC outperformed on average CLWC in all created scenarios for synthetic data, also obtaining a lower standard deviation in all scenarios, showing the stability of the algorithm. Regarding the real data, three data sets from UCI were augmented with 25 noisy features. 3SMBC obtained also very competitive results, outperforming CLWC on average in most situations taking into account three different inputs for the number of clusters. When the correct number of clusters was fixed to CLWC, 3SMBC obtained better results on average when there were 40% labeled instances and one cluster was hidden in the input data. Besides, the standard deviation results for CLWC were again higher in general,

due to the different initializations, than for 3SMBC. Note that neither the correct number of clusters nor a validation for the initialization are usually known parameters in this kind of problems.

Related to soft clustering, it is interesting to comment the possible relation between soft clustering, projected clustering, and subspace clustering. 3SMBC is based on soft clustering, therefore, and in theory, some overlapping is allowed in the found clusters, since an instance may belong to different clusters with different probabilities. The overlapping in clusters is allowed in subspace clustering but not in projected clustering. However, due to the almost hard clustering results obtained, and even more to the post-process step, 3SMBC can be considered as a projected clustering algorithm.

Part IV

APPLICATIONS IN NEUROSCIENCE

Chapter 6

Introduction to neuroscience

6.1 Introduction

Neuroscience is the science that deals with the study of the nervous system. Although this discipline could be considered very far from computer science, this distance is becoming shorter nowadays. Some computer science approaches, such as data analysis and visualization, are very often used in biological research. These common endeavours between different fields lead to interdisciplinary tasks that allow getting advances in challenging tasks related to the involved disciplines.

Biology in general, and neuroscience in particular, have been very typical fields in which data mining techniques obtained good results in the past. Then, and once the data gathering is moving forward with advances in both storage and processes, the union of traditional machine learning techniques together with new advances in this area, such as the created in this thesis, must be employed to continue obtaining good results in this kind of interdisciplinary tasks.

More specifically, the task presented as real application is, roughly speaking, **classification of neurons**. This is a very challenging task in neuroscience, since many efforts have been done for many years, trying to reach some consensus among experts. However, the agreement has not been reached up to date.

Chapter outline

This chapter continues in Section 6.2 providing some history about neuroscience. Some useful definitions and concepts, which are necessary to understand the real applications that will be presented in next chapters, are also introduced. Sections 6.3 and 6.4 present an European project and its Spanish participation, respectively, both focusing on the study of the brain. Finally, the current problem statement is detailed in Section 6.5.

6.2 History

It is necessary to go back to the 18th century to find the inception of modern neuroscience. During the 1780's, Luigi Galvani, an Italian biologist, physician, and anatomist, researched about electrical charges [97]. Throughout that experimentation, Galvani discovered the current electricity. This was one of the milestones for modern neuroscience, because electricity is very related to the human beings capacitance. After this discovery, the next main step in neuroscience was given by Camilo Golgi, an Italian physician and cytologist. Golgi discovered a novel technique for staining nervous tissue in 1873 [106]. The technique was called *reazione nera* (*black reaction*) and consisted of impregnating the preparation, previously hardened in potassium bichromate, using silver nitrate. This method allowed to track the cell ramifications in a manner that had not been possible previously (see Figure 6.1). The methods of staining before the introduction of the *black reaction* (mostly carmine staining) only allowed visualization of neuronal cell bodies and a small portion of their proximal processes, making further characterization of cells difficult. The complete staining of the cell, that is, with all its parts, and its finest morphological details were readily observed in Golgi-stained preparations. At this time, the most common view regarding the organization of the nervous system was that it consisted of a diffuse network of nerves formed by the anastomosing branches of nerve cell processes (*reticularism*). Therefore, experts from this branch, including Golgi, argued that the nervous system was a continuous network.

After some years, Santiago Ramón y Cajal, a Spanish physician and scientist, not only used the *black reaction* but also adapted it in his research [218, 219, 222, 223, 224]. Cajal's studies of the microanatomy of virtually the whole central nervous system and his observations regarding degeneration and regeneration, together with his theories about the function, development, and plasticity of the nervous system, had a profound impact on researchers of his era. These studies represent the roots of what some of the most exciting areas of discovery, in terms of the structure and function of the brain in both sickness and health [67], are nowadays. For these reasons Cajal is considered the father of modern neuroscience. Cajal initialized a new branch, distinct of the *reticularist*, in which cells, called neurons, were considered as independent and basic units and their connections are by contiguity, instead of continuity. This was called the *neuron doctrine* [217] and was one of the most important milestones in neuroscience. Based on this doctrine, Cajal's studies with the Golgi method confirmed Golgi's conclusion that dendrites end freely and do not anastomose. However, in contrast to Golgi,

Figure 6.1: Cerebellar cortex image using the *reazione nera* technique, showing relations between two different types of cells [178].

Cajal added the crucial conclusion that this also applies to axons and their branches, and provided many examples from throughout the nervous system to support this hypothesis [67]. An original draw, as an example of Cajal's work about the types of cortical neurons, can be seen in Figure 6.2. As a curiosity, Golgi and Cajal shared the Nobel prize for medicine in 1906, due to their important and, in many cases, opposite, research in the field.

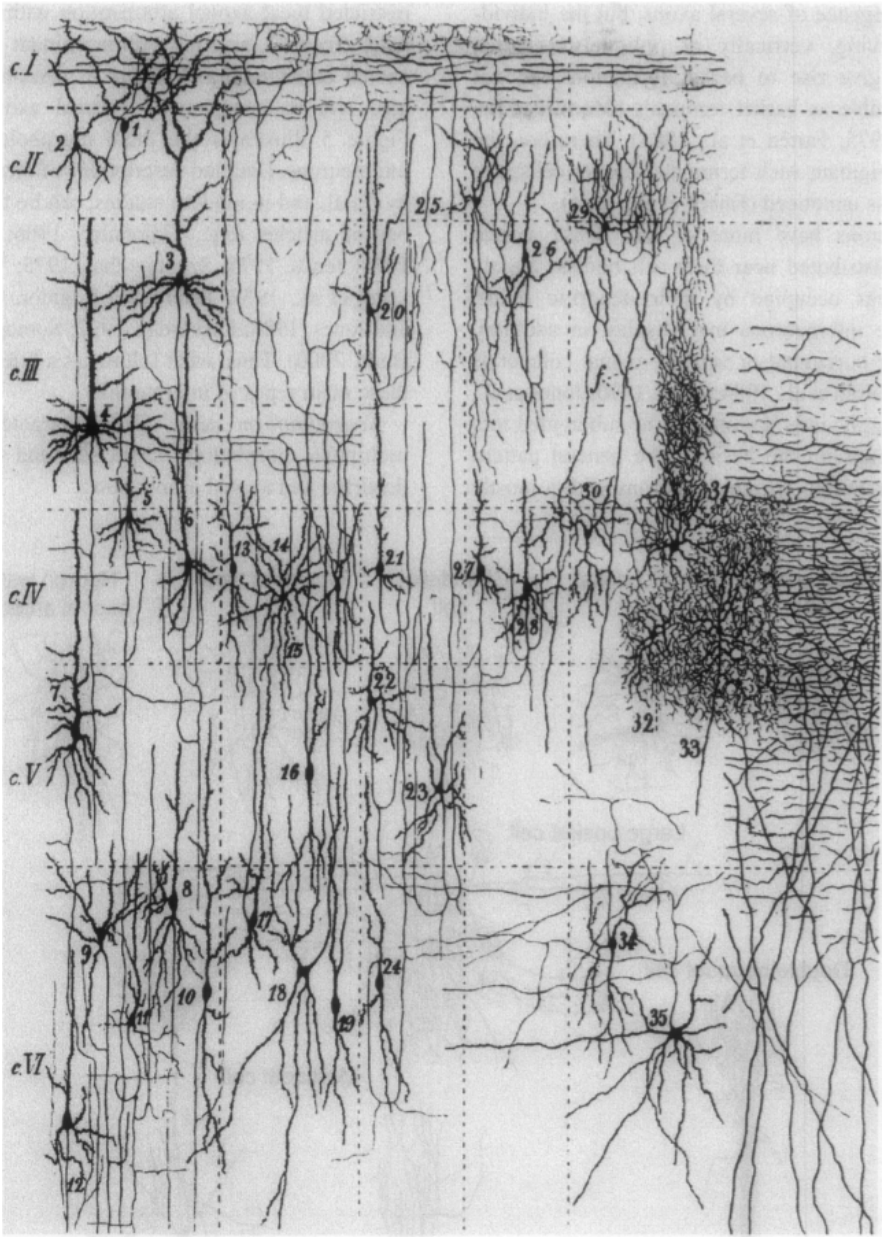


Figure 6.2: Cajal's drawing, extracted from [134], showing different types of cortical neurons and unmyelinated axons.

Neuron doctrine

The *neuron doctrine* is the fundamental principle of modern neuroscience [41]. Before detailing its assessments, it is interesting to know that the name of neuron was given to the basic structural unit in the brain by Waldeyer [257].

The *neuron doctrine* established the neuron as the structural and functional individual unit of the nervous system. This discovery was controversial because the theory of the *neuron doctrine* was completely opposite to the belief of the *reticularist* view, which said that the human brain was a complex and continuous network. However, according to the *neuron doctrine*, neurons are not anatomically continuous to other neurons, and therefore neuron connections are not made by continuity anymore, but by contiguity.

Neurons can be divided into three different parts:

- Soma. It is the cell body, contains the nucleus, and stores the genetic information of the cell. The soma gives rise to the two other parts of a neuron.

- Axon. It is the transmitting element of neurons. Its length can be very different depending on the neuron type and is very thin compared to the soma. The axon has several terminal arborizations, which make close contact to dendrites or the soma of other neurons. Therefore, the connections between neurons, called synapses, start in the axon of a presynaptic neuron and signals are transmitted to another neuron (postsynaptic cell).

- Dendrites. If the axon is the output element of a neuron, dendrites are the input elements. They are multiple short branches classified into apical and basal dendrites, depending on whether they emerge from the apex or the base, respectively. Some dendrites have short extensions, called spines. These structures represent the main postsynaptic element of cortical synapses. The presence of spines was also discovered by Cajal [217] and confirmed in [220, 221]. The importance of spines is pointed in [100]. Figure 6.3 shows an original drawing made by Cajal with many spines in neuron dendrites.

Figure 6.3: Original drawing made by Cajal representing neurons with many spines from Instituto Cajal, Consejo Superior de Investigaciones Científicas (CSIC), Madrid, Spain.

Many morphological features can be extracted from each part of a neuron. The application of machine learning to neuroscience in this thesis is focused on this kind of data. Nevertheless, many other important features, but out of the scope of this work, like electrophysiological and genomic ones could also be gathered.

This work is focused on the cerebral cortex, which is a structure directly involved in many aspects of mammalian behavior and considered the most human part of the nervous system. A brief introduction to the history about the morphological types of neurons in the cerebral cortex follows.

Types of neurons

Before the discovery of the Golgi method, the existence of two broad morphological types of cortical neurons was already recognized: pyramidal and nonpyramidal neurons [142]. However, it was with the introduction of the Golgi method when the complete staining of the neuron and its finest morphological details were readily observed in Golgi-stained preparations. This led to the important breakthrough of full characterization and classification of neurons.

Golgi distinguished then, according to the behaviour of the nerve process, between *Type I* and *Type II* cells. Cells from *Type I* had a motor function and, morphologically, were cells with long axons. The long axons are important because these cells were able to form long circuitry going even beyond the grey matter. On the other hand, cells from *Type II* had a sensory function, short axons, and were in charge of local circuits. Cajal agreed basically with the morphological aspect and distinguished between long-axon and short-axon cells. Since then cells with long axons have been considered as pyramidal neurons due to their roughly triangular soma. Another important characteristic of pyramidal cells is the presence of apical dendrites, besides the basal dendrites. Finally, according to the role in connections, these neurons are usually excitatory, using glutamate as neurotransmitter. This type of neurons is the largest group in the cerebral cortex and constitute 70-85% of the total neuron population. On the contrary, nonpyramidal neurons are today subdivided into two large groups: spiny nonpyramidal or stellate cells, and aspiny or sparsely spiny nonpyramidal cells. Spiny nonpyramidal cells form a morphologically heterogeneous group of neurons, and some of them project to other cortical areas, whereas others are short-axon cells, also called interneurons. Aspiny nonpyramidal cells are interneurons that constitute the majority of short-axon cells and approximately 15-30% of the total neuron population. These neurons are usually inhibitory, using GABA as neurotransmitter, and show a great variety of morphological, biochemical, and physiological types. This basic separation is widely accepted by experts, and although some subgroups have been found as expansions to the basic classification, differences in nomenclature and neuron types do not allow to create a good and complete neuronal classification.

This is a very basic introduction to some aspects in neuroscience that will be useful for the real application part of this thesis. Nevertheless, the reader can refer to the first chapters of [144], as an extended introduction for a better understanding of neuroscience terms.

Many efforts are being done by different research groups all over the world, existing narrow relations between data analysis and neuroscience. Specifically, the study of the basic functional unit of the brain, the cortical column, is crucial for the basic understanding of the brain function. Examples of some research efforts are presented next.

6.3 The Blue Brain Project

The Blue Brain Project (BBP) [185] is an example of one of these projects aiming to research the brain. The BBP started in 2005 at the École Polytechnique Fédérale de Lausanne (EPFL) as the first large scale attempt to carry out the reverse engineering of the human brain at all levels of detail, from genomics and molecular interactions to cognitive processes. The BBP is focused nowadays on the *cortical column hypothesis*.

The *cortical column hypothesis*, formulated by Mountcastle [198, 199] after his research about the cortex of cats, is related to the organization of the cerebral cortex. According to the *cortical column hypothesis*, the neocortex is made up of many elemental units, called cortical columns, which are repeated along cortex surface with small variations. There is some accepted knowledge about cortical columns, such as each column is divided into six horizontal layers, with different neuron types and connections.

In the first phase of the BBP, and having gathered data from experiments with rats for 15 years, a first circuit model with 10,000 neurons corresponding with a cortical column, was simulated. The ultimate goal of this project is to provide a computational model of the brain, to enable simulations at all possible levels of detail. To achieve this goal, more realistic circuits and simulations about the cortex must be provided not only by gathering more data from experiments but also by improving visualization and data analysis techniques that can deal with those data. It is obvious the complexity of the brain cortex from Cajal's time, so obtaining accurate models of the cerebral cortex is a good foundation stone to reach the final goal about the whole brain.

6.4 The Cajal Blue Brain Project

The Spanish participation within the BBP is called Cajal Blue Brain Project (CBBP). Different universities and laboratories, such as the Universidad Politécnica de Madrid (UPM) and the Instituto Cajal (IC) from Consejo Superior de Investigaciones Científicas (CSIC), respectively, are involved in the brain model aim. Although the participation of the UPM with the BBP goes back to the origin of the international project, the CBBP was officially initiated in 2008.

Both the BBP and the CBBP are involved in different tasks to accomplish the final goal of the projects. Many of these tasks are related to visualization (Figure 6.4 shows an example of a related visualization work) and data analysis. The real application of this thesis is focused on the latter. Within the data analysis tasks, the classification of neurons from the cerebral cortex is necessary to get an accurate brain model. Hence, to understand neural

circuits it is fundamental, as a first step, to correctly identify the existing types of neurons. If this classification were achieved, other knowledge, like the connections and how the circuit actually works, would be more easily researched.

6.5 Problem statement

The classification of neuron types based on morphological features is a challenging task in neuroscience. This is a very open problem and agreement among experts has not been achieved up to date. However, there is some knowledge about the different types and about the most important features to characterize them. Thus, this is valuable information that should be use to accomplish the classification.

The top level of the classification, separating between pyramidal neurons and interneurons, seems clear for experts, but has not been automatically obtained previously. The study shown in Chapter 7 aims at separating pyramidal neurons from interneurons. This is a noteworthy study not only to obtain an automatic separation between the two types of neurons, but also to introduce some supervised algorithms to neuroscience. This introduction is useful because these techniques have not been used very often in this domain, even when they were the most suitable approaches regarding the input data and the purpose.

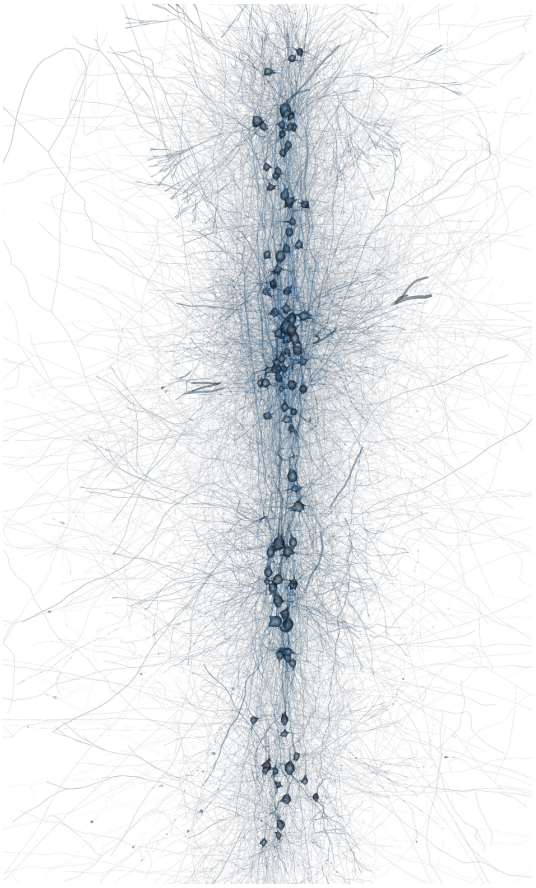


Figure 6.4: A partial view of a synthetic cortical minicolumn made by 100 neurons, extracted from [126].

Once the separation between pyramidal neurons and interneurons (the first level of the classification) is achieved, further studies are necessary to move forward towards a more detailed classification. These movements are commonly related to interneurons, as can be seen in an important attempt [9] that tried to reach some consensus about the different types of interneurons. Following this direction, the proposal presented in Chapter 5, together with a supervised algorithm, are used in Chapter 8 to throw some light about different types of interneurons in the cerebral cortex.

Chapter 7

Pyramidal neurons vs interneurons

7.1 Introduction

The two principal neuronal types of the cerebral cortex are pyramidal neurons and GABAergic interneurons [212, 222] (see Figure 7.1). This basic classification has been expanded over the last century with the definition of new subtypes of cells, but the agreement related to these expansions is rather limited. At the same time, classification of cortical neurons has traditionally been qualitative [66] with nomenclature that varies across investigators. For these reasons, it has become apparent in recent times that a classification based on quantitative criteria is needed, in order to obtain an objective set of descriptors for each cell type that most investigators can agree upon. As suggested by community efforts [9], proper neuronal type definition should probably be a multimodal information task, including physiological, molecular and morphological features, and should use classification algorithms that are both quantitative and robust [43]. However, the data acquisition is not a trivial task, and the number of reconstructed neurons characterized by morphological features is not very high (even taking into account many different laboratories all over the world). This number becomes even lower if neurons characterized by physiological or molecuar features are desirable. Thus, all the neurons used in this and the next chapters are characterized only by morphological features.

Previous efforts to quantitatively classify cortical neurons have based their neuronal classification on clustering techniques [23, 43, 77, 123, 124, 125, 145, 158, 248, 264]. As explained in Section 2.3, these are essentially exploratory techniques which aim at discovering new subtypes of cells or confirming some known hypothesis about them. In these studies, prior information on the potential outcomes was not utilized, or was only used to validate the clustering. Instead, this information could be used to guide a supervised classification (see Section 2.2). An example of this approach can be seen in [184], where linear discriminant analysis was used to investigate whether different classes of projection neurons had distinct axon projection patterns. This problem was also tackled by Wong et al. [264], using hierarchical clustering.

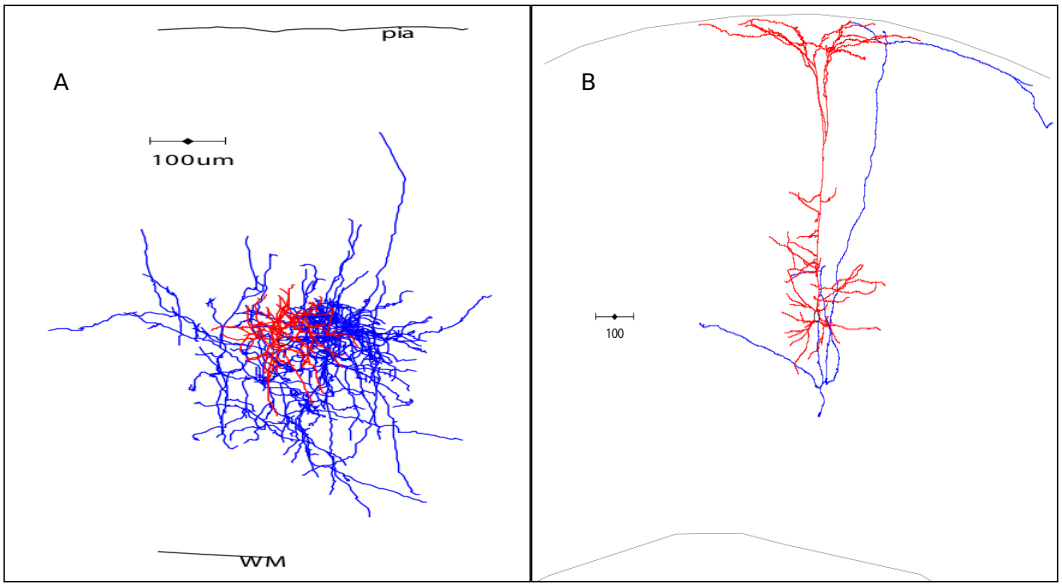


Figure 7.1: Picture of 3-dimensional reconstructions of an (A) interneuron and a (B) pyramidal cell from the Rafael Yuste's lab at Columbia University.

In this study, and as the first level of a classification of neurons, the performance of supervised and unsupervised classification approaches is compared in an apparently simple task: to automatically distinguish pyramidal neurons from interneurons. It is important to note that, in this exercise, the presence or absence of an apical dendrite was not included in the morphological features, since it was used as the ground truth to evaluate the performance of the algorithms. This work, presented in [113], has two main goals:

- To automatically separate pyramidal neurons from interneurons using machine learning techniques, establishing the first level of a classification of neurons.

- To introduce supervised classification techniques into neuroscience, as an approach that must be used when the aim is to classify instances into known classes, and there are available labeled training data. Besides the classification algorithms, different techniques to automatically select the most suitable subset of features are also evaluated and compared with the most widely used techniques in neuroscience up to date.

Chapter outline

After the introduction, the chapter continues with the experimental results in Section 7.2, which covers all the used data, the evaluation process, the obtained results, and the comparison of algorithms. The summary of this study and some discussion about it are presented in Section 7.3.

7.2 Experimental results

7.2.1 Data

The used data set has 327 cells (199 interneurons and 128 pyramidal neurons), and for each cell, 65 morphological features were measured, creating a data matrix. All the neurons were reconstructed from brain slices of the cortex of mice. Neurons were reconstructed with an advanced scientific software for brain mapping, neuron reconstruction, anatomical mapping, and morphometry, called Neurolucida (MicroBrightField). Once the reconstructions were obtained, the neurons were measured with Neurolucida Explorer to extract their morphological features. Some variables were directly measured, such as somatic area and perimeter, number of axons and dendrites, axonal and dendritic length, axonal and dendritic branch angles and number of axonal and dendritic nodes (branch points). Other variables were calculated values such as axon and dendritic sholl lengths, convex hull analysis and fractal analysis. Sholl length is a measure of how the length of the processes is distributed. Concentric spheres centered at the soma were drawn around the neuron; for axons the spheres were drawn at radius intervals of 100 μm and for dendrites at intervals of 50 μm. The sholl length is the total length of the part of the axon or dendrite contained within in each shell. Convex hull analysis draws a convex shape around the axons or dendrites in both two (x,y) and three (x,y,z) dimensions. The area and perimeter of the two dimensional shape and the volume and surface area of the three dimensional shape are then calculated. Fractal analysis calculates the fractal dimension of the axons or dendrites using linear regression, and thus is a measure of how the neuron fills space. All pyramidal neurons had apical dendrites. Interneurons belonged to many different subtypes and were collected over several different studies from the Rafael Yuste's lab at Columbia University.

7.2.2 Evaluation process

The evaluation process aims to compare previously used algorithms for classification in neuroscience, i.e., hierarchical clustering, with supervised classification algorithms. Besides the learning algorithms, different dimensionality reduction techniques are also compared.

Agglomerative hierarchical clustering, based on Euclidean distance and Ward's method as linkage criterion (see Section 2.3), is used as clustering algorithm. The final number of clusters is always 2, since it is a binary classification problem, therefore this value is known beforehand. On the contrary, different supervised classification algorithms, explained in Section 2.2, are used trying to cover some of the most important paradigms: NB, C4.5, K-nn, MLP, and LR. The measure of both types of classification performance is the rate of correctly classified instances or accuracy, using the presence or absence of an apical dendrite as the ground truth. The supervised validation technique is 10-fold cross validation (see Section 2.2.2). Regarding the dimensionality reduction techniques, PCA is used as FSE algorithm, whereas a filter and a wrapper approaches, using three different search techniques: forward selection, backward elimination, and genetic algorithms, are used as FSS techniques. See Section 2.5 for details

about the algorithms and techniques. Finally, the results of the supervised algorithms are compared using the Wilcoxon signed-rank test, previously used in Section 4.4.4.

7.2.3 Results

The obtained results, using the different techniques listed above, are:

Hierarchical clustering

This approach was used with three different dimensionality reduction techniques. The first one was based on the first six PCs obtained with PCA, which carry almost 55% of the total variance. This number of PCs was chosen because of the trade-off between the accuracy and the number of features. For example, using the first seven PCs (60% of the total variance), the accuracy decreased by 2%. Using the first eleven PCs (70% of the total variance), the accuracy was only increased in 1%. And finally, using the first 16 PCs (80% of the total variance), the accuracy decreased in 4%. The second variable selection method for clustering was to use only those original features with a correlation coefficient greater than 0.7 with the first six PCs. With this requirement, 10 original features remained. Finally, filter FSS was used as the third method to select variables in unsupervised approach.

As it was known beforehand which neurons were pyramidal and which were interneurons, the accuracy of the hierarchical clustering was calculated as the percentage of each group of cells which fall in the correct majority cluster, after separating the data into two final clusters. Thus, it is assumed that each cluster was equivalent to a class.

		Accuracy (%)	F
No FSS		59.33	65
PCA	PC	59.02	6
	Original features	66.77	10
Filter	Forward	77.68	10
	Backward	71.25	17
	Genetic	79.82	16

Table 7.1: Accuracy results obtained with hierarchical clustering. PC uses the six first principal components, whereas "Original features" uses the original features with correlation greater than 0.7 with the six first principal components. The number of features used (F) is also indicated.

All the hierarchical clustering results can be seen in Table 7.1. Without dimensionality reduction techniques, 59.33% of accuracy is obtained. Using the above techniques of dimensionality reduction related to PCA the outcomes were relatively poor. Only 59.02% accuracy was reached using PCA, which is the lowest value from all algorithms in this comparative study. Using hierarchical clustering of the more than 0.7 correlated features with the PCs, the accuracy obtained was 66.77%. This is increased when the features obtained with filter FSS were used. The accuracy obtained is 71.25% using backward elimination, and this value increased to 77.68% using forward selection and 79.82% using genetic algorithms.

As mentioned, all these accuracy values were obtained without using any previous information about the class variable. Supervised classification algorithms, whose results are

presented next, use this known information to build the different models.

Supervised algorithms

A battery of different supervised classification algorithms were compared in the task of distinguishing between pyramidal neurons and interneurons. Again, all the available data is used first, without FSS. Filter FSS was then used with three different search strategies, the same as with hierarchical clustering. Finally, wrapper FSS, another approach used to select subsets of features which is only appropriate for supervised classification algorithms, is explored. Thus, a comparison using it with clustering techniques cannot be made. In each table, the values correspond to the accuracy of each model, i.e. the mean ± standard deviation (percentage) averaged over the 10 values estimated using 10-fold cross-validation. The number of features used is also indicated as before. Bold face indicates the model with no significant statistical differences with the highest accuracy supervised model.

Naïve Bayes This algorithm obtained very similar results using all variables and using variables selected by the filter FSS process (see Table 7.2). Without FSS, an 80.73% ± 10.44 accuracy was achieved, whereas with filter FSS, the accuracy was around 80%. Wrapper FSS was able to improve these means: with forward search, its accuracy was 87.16% ± 6.34. Backward (83.18% ± 9.12) and genetic search (83.49% ± 8.55) did not significantly improve the accuracy.

		Accuracy (%)	F
No FSS		80.73 ± 10.44	65
Filter	Forward	79.82 ± 9.86	10
	Backward	79.51 ± 9.74	17
	Genetic	80.43 ± 7.07	16
Wrapper	Forward	**87.16 ± 6.34**	8
	Backward	83.18 ± 9.12	50
	Genetic	83.49 ± 8.55	23

Table 7.2: Results obtained using Naïve Bayes (see the text for bold face meaning).

C4.5 In the case of C4.5 algorithm, all the results (see Table 7.3) were comparable or better than those obtained using naïve Bayes. Without FSS, an 84.40% ± 3.84 of accuracy was obtained. Forward selection and genetic algorithms for filter FSS showed lower outcomes than without FSS, but by using backward selection a performance of 88.07% ± 6.09 using only 11 features was achieved. This mean was the highest one obtained using filter FSS. In the

		Accuracy (%)	F
No FSS		84.40 ± 3.84	65
Filter	Forward	82.26 ± 7.17	9
	Backward	**88.07 ± 6.09**	11
	Genetic	81.65 ± 7.24	6
Wrapper	Forward	86.85 ± 5.29	7
	Backward	**87.16 ± 5.83**	12
	Genetic	**86.85 ± 4.72**	13

Table 7.3: Results obtained using C4.5

case of wrapper FSS, the outcomes were 86.85% ± 5.29 using forward selection, 87.16% ± 5.83 using backward selection and 86.85% ± 4.72 using genetic search.

K-nn K-nn was configured with K = 5 after trying some preliminary tests, this configuration obtained better accuracy than K = 1, K = 3, and K = 7. In spite of K-nn being the simplest algorithm used to classify, the results (see Table 7.4) were quite competitive with other approaches. Specifically, with K-nn using all the available variables a 83.18% ± 7.15 accuracy is obtained. This value improved when filter FSS is used, obtaining 85.01% ± 5.60 with genetic algorithms as the best case. Again, wrapper FSS was the best approach to select appropriate variables, with accuracies using backward selection of 86.85% ± 6.26, and in turn, this is overcome by 87.46% ± 5.68 for genetic algorithms and 89.30% ± 7.58 for forward selection.

		Accuracy (%)	F
No FSS		83.18 ± 7.15	65
Filter	Forward	83.79 ± 9.55	10
	Backward	84.71 ± 6.03	17
	Genetic	85.01 ± 5.60	16
Wrapper	Forward	**89.30 ± 7.58**	6
	Backward	86.85 ± 6.26	51
	Genetic	**87.46 ± 5.68**	34

Table 7.4: Results obtained using K-nn, with K = 5

Multilayer Perceptron Multilayer perceptron (see Table 7.5) was the algorithm with the highest overall accuracy among all the algorithms without using FSS (87.46% ± 9.06). Moreover, this result was improved using backward selection for filter FSS (87.77% ± 6.36). However, using forward selection (82.57% ± 9.54) or genetic algorithms (82.26% ± 9.17), the accuracy was reduced. The improvement obtained using wrapper FSS was not as significant as when using other supervised algorithms. In this case, 88.07% was the highest accuracy mean obtained using forward selection (± 4.99) and backward elimination (± 8.27).

		Accuracy (%)	F
No FSS		**87.46 ± 9.06**	65
Filter	Forward	82.57 ± 9.54	10
	Backward	82.77 ± 6.36	17
	Genetic	82.26 ± 9.17	16
Wrapper	Forward	**88.07 ± 4.99**	10
	Backward	**88.07 ± 8.27**	61
	Genetic	87.46 ± 6.26	37

Table 7.5: Results obtained using multilayer perceptron (MLP)

Logistic Regression The last supervised classification algorithm, logistic regression (see Table 7.6), maintained the mean obtained without FSS (82.26% ± 7.36) when forward selection for filter FSS was used (82.26% ± 9.82). This outcome is 83.49% ± 9.45 using genetic algorithms while using backward elimination reaches 85.63% ± 8.56. The highest accuracy of all the approaches was obtained using logistic regression with wrapper FSS and a genetic

		Accuracy (%)	F
No FSS		82.26 ± 7.36	65
Filter	Forward	82.26 ± 9.82	10
	Backward	85.63 ± 8.56	17
	Genetic	83.49 ± 9.45	16
Wrapper	Forward	**85.63 ± 9.79**	7
	Backward	84.71 ± 7.54	59
	Genetic	**91.33 ± 5.95**	33

Table 7.6: Results obtained using logistic regression (LR)

algorithms search: 91.13% ± 5.95. This model was therefore used in the statistical test to be compared against the rest.

7.2.4 Algorithms comparison and feature relevance

For hierarchical clustering, filter FSS always generated more accurate classifications than using all available variables, or after applying some traditional dimensionality reduction technique such as PCA. It is important to highlight this result because all previous clustering works used PCA to reduce the number of variables. Specifically, for this test, using filter FSS enhanced accuracy of unsupervised clustering by almost 15%. Thus, this approach appears desirable to select an appropriate subset of variables for future cluster analysis studies.

When comparing hierarchical and supervised methods, hierarchical clustering and filter FSS, using forward selection or genetic algorithms, were competitive combinations against supervised classification algorithms with no FSS and filter FSS. On the other hand, when wrapper FSS is used with the supervised classification algorithms it is generally superior.

After concluding that supervised methods with wrapper selection of variables enhance the classification, the next step was to discern which supervised algorithm was best able to discriminate between pyramidal neurons and interneurons in this test.

The highest accuracy was obtained using the model built with logistic regression and wrapper FSS (with a genetic algorithm). To compare this model with all the rest, the Wilcoxon signed-rank test was used.

The results obtained with this statistical test are shown in Table 7.7. In this table, only the models which have a p-value greater than 0.05 (differences are not statistically significant) in the test are shown. As these models did not reject the null hypothesis, it cannot be asserted than they are significantly different from the model built using logistic regression and genetic algorithms in a wrapper approach. Thus these models are the top models from these results. Statistical hypothesis test outcomes confirm that models obtained with the wrapper approach are the most accurate to classify interneurons and pyramidal neurons, since nine of the selected models in Table 7.7 are built using wrapper FSS. These results indicate that there is not one particular supervised method which is superior, since all the used algorithms could be chosen as winners based on the statistical test. Therefore, an appropriate selection of variables (using wrapper FSS in this case) appears to be more

important than using a specific supervised algorithm.

FSS		Algorithm	p-Value
No FSS		MLP	0.091
Filter	Backward	C4.5	0.095
Wrapper	Forward	NB	0.095
		K-nn	0.220
		MLP	0.063
		LR	0.053
	Backward	C4.5	0.077
		MLP	0.115
		C4.5	0.052
	Genetic	K-nn	0.052
		LR	-

Table 7.7: Models which do not reject the null hypothesis, and therefore, with no significant statistical differences (p-value greater than 0.05) with the highest accuracy model are listed.

Finally, which of the actual morphological features, or combinations of them, were most indicative of differences between pyramidal neurons and interneurons, is studied. In the original data set, 65 variables were available before applying subset selection. When filter FSS was applied, the number of attributes obtained for each searching method was the same, except for C4.5 algorithm. This is because filter FSS algorithms do not depend on the classification method to obtain the subset of features. The number of features selected, using filter FSS, was 10 for forward selection, 17 for backward selection, and 16 for genetic algorithms. C4.5 is the only algorithm with different number of features, since it has an embedded FSS that chooses a subset from the features selected by the filter FSS to build the decision tree.

The number of features selected using wrapper FSS were similar but the main difference was in the searching technique. Using forward selection, the number of features selected was in a range from 6 to 10. The low number of features is a bias of the forward selection. In the case of backward elimination, the number of features was higher (from 50 to 61) with an exception in the C4.5 algorithm. In C4.5, the number of features selected by the wrapper FSS was 23, and after that, when C4.5 induces the decision tree model, only 12 features were used. Genetic algorithms technique selects from 13 to 37 features, taking into account again that C4.5 has the embedded FSS. This technique was not as biased as the two others, since it is not a greedy search.

Regarding the particular features chosen, the somatic compactness seemed to be the most important feature among the somatic features because it was the most commonly selected variable by the winner models. As for axonal features, the number of axonal sholl sections and standard deviation of the average axonal segment length were the two most important features. This happens in the logistic regression and C4.5 models for example, because these two features had a high coefficient or are located at the top of the tree. In addition, the axonal local angle average was another important feature because it was selected by many models. For the same reasons, the number of dendritic sholl sections and the ratio of dendritic length to surface area were the most important dendritic features. The highest order dendritic segment is selected by the majority of the models as well.

Some tests, using separately the somatic, axonal and dendritic subsets of features, were

also performed on some of the selected models. While models built using only somatic features obtained ∼60% accuracy, ∼75% accuracy was obtained with axonal features while dendritic features reached ∼85% accuracy. These values confirmed the importance of dendritic features. Therefore, these results indicate that dendritic features are very informative to differentiate morphologically pyramidal neurons from interneurons, although some axonal and somatic features can also contribute to this distinction.

7.3 Summary and discussion

To enable the quantitative classification of neuronal cell types, different methods were compared in this methodological study to objectively distinguish between neuronal classes, based on their morphologies. By using a standard database with a clearly classified set of cells, a test in which the algorithms had to distinguish pyramidal neurons from interneurons is created. A human observer originally classified these cells into both classes according to the presence or absence of an apical dendrite, thus setting the ground truth for this task. Then the performance of the unsupervised clustering method, which is becoming standard in neuroscience, was tested side by side, versus the performance of representative algorithms from some of the most popular supervised classification paradigms used in machine learning. The reason for doing so is that, if previous information is available to classify data, taking advantage of it to obtain more accurate outcomes should be desirable. Nevertheless, given the peculiarities of the classification problem, it was not obvious that those supervised methods would be in principle better than previously used neuronal classifiers, or which approach could outperform the others, so the task of carefully comparing a battery of algorithms and different preprocessing strategies was undertaken.

The main finding is that supervised classification methods outperformed unsupervised algorithms. In this comparative study, hierarchical clustering approach was unable to obtain an accuracy as precise as supervised classification when distinguishing between pyramidal neurons and interneurons. Therefore, supervised classification is an effective approach to perform this task and is another approach in neuronal data analysis, something that could be useful in future studies. In fact, previous classification studies, in which some information is known beforehand, could be reanalyzed using that information as a class label with supervised algorithms. An ideal supervised classification algorithm does not emerge from these results. It seems that the accuracy of results obtained does not depend on the classification algorithm, since the best models chosen using the statistical test are built using all the different supervised classification algorithms tested. Thus, the choice of the algorithm would depend on each specific classification or domain. There could be some bias in this choice, since if an interpretable model is desirable, C4.5 or naïve Bayes could be the most preferred. K-nn does not build a model, so this could be an undesirable restriction.

The second conclusion is that the preselection of the variables with FSS greatly enhances the performance of both supervised and unsupervised methods. Specifically, in terms of which FSS approach to follow, wrapper FSS is found as the most suitable technique for this data

set of neurons using supervised algorithms. Models obtained using FSS are desirable, not only because a higher accuracy is achieved, but also because more parsimonious and easily understood models are obtained. The disadvantage of this approach is its computational cost, since performing wrapper FSS is slow. Wrapper FSS cannot be used with unsupervised algorithms, but the results obtained using a different variable preselection method, the filter FSS, with hierarchical clustering point out the advantage of using this dimensionality reduction technique, compared to clustering with no FSS.

The final conclusion is that an acceptable distinction between pyramidal neurons and interneurons was achieved using dendritic morphological features, even without explicitly providing knowledge of the presence or absence of an apical dendrite. But while differentiating between pyramidal neurons and interneurons may not seem a particularly difficult task for a trained neuroanatomist, discerning subtypes of neurons using objective and quantitative criteria is more challenging. Therefore, one future direction could be the quantitative exploration of new subtypes of interneurons. For this goal, unsupervised clustering techniques could still be used as exploratory techniques; to accomplish this goal, a priori information will probably be most useful, or even key. For this task, one could explore the use of semi-supervised clustering, using previous information about cell groups that are well accepted in the scientific community, as a way to partially supervise the clustering.

Although it is difficult to reach a consensus about cell types that exist in the cortex, the introduction of supervised, or partially supervised algorithms could help to advance the state of this key question, an essential one to decipher neocortical circuits. For this reason, 3SMBC, one of the proposals of this thesis, is applied to throw some light in the classification of interneurons (see Chapter 8).

Chapter 8

Subtypes of interneurons

8.1 Introduction

The first level of a possible classification of neurons was studied in the previous chapter. That level is commonly accepted nowadays, but new levels of the possible hierarchical classification must be studied. The most representative efforts related to a possible next level have been done with interneurons [9]. However, the scientific community still lack a catalog of neuron types and names that is accepted by the general scientific community. A milestone towards a future classification of interneurons in the cerebral cortex was the standardization of the nomenclature of their properties [9]. Nevertheless, at that time it was not possible to identify a set of anatomical traits that unambiguously define an interneuron class.

The final goal is to identify subtypes of interneurons according to morphometric measures extracted from 3-dimensional reconstructions. The main motivation for this task is that, once a classification is achieved and accepted, it may be easier not only to tackle other related tasks, like brain modelling, but also the understanding among experts, since interneuron nomenclature would be unique for all laboratories.

The classification of neurons is a known problem since Cajal's time. Therefore the first attempts to accomplish the task began more than one hundred years ago. Taking into account that the task has not been successfully solved yet, it can be considered as a very challenging task. From engineer and computer science points of view, there are two main reasons that make this task hard to solve:

- Lack of data. Biological data are commonly difficult or expensive to gather. Neuronal data are not an exception. The neuron reconstructions are used to extract, using specific software, morphometric measures that describe each neuron. But it is not easy to obtain reconstructions that represent the total extension of each neuron. Besides, the number of correctly reconstructed neurons is not necessarily representative of all possible subtypes of neurons. This last fact should not be a limitation, since if an accepted classification is achieved with the available data, the obtained milestone would be a big enough success even although the classification is not complete.

- Lack of agreement. This can be considered a problem according to two different aspects. The first one is related to take advantage of some available knowledge, since some knowledge for an expert could not be the same knowledge for another expert. However, not all the subtypes of interneurons are equally unknown, and, therefore, any available knowledge should be used to improve and guide the classification. The second aspect is that any obtained results must be validated by experts at last. Thus, validation can be also challenging if experts have different previous opinions about the subtypes.

This chapter presents a study related to the classification of interneurons assuming the complexity of the task, due to the reasons commented above. First of all, the number of available neurons is limited, but this number is even more reduced depending on if labeled neurons must be used. Also regarding the available data, the reconstructions cannot be equally made for every neuron. This is a recurrent problem that is assumed and, therefore, results must be taken wisely. On the other hand, the limitation about the agreement on types of neurons is also tackled during the chapter.

In particular, a known supervised classification algorithm and the proposal presented in Chapter 5 are used for the classification task. Then, the procedure to be followed and the results are validated by an expert neuroscientist who is working with the IC in the CBBP.

Some considerations about nomenclature for this chapter are: since the chapter focuses only on interneurons, the word "neuron" refers to interneuron unless otherwise indicated. As detailed in Section 8.3.1, some types of interneurons are proposed for the experiment. Thus, the words "type" and "group" refer to each of these types of interneurons, whereas the words "subtype" and "subgroups" refer to possible new subpopulations of neurons.

Chapter outline

The following section describes a recent experiment that tested the suitability of a set of terms as the basis for anatomical classification of cortical interneurons. The results obtained in this experiment have been used to obtain the labels for the data that are presented in this study. Section 8.3 details these data, together with the procedure and algorithms used in the new study. Finally, Section 8.4 covers a summary of this study and some discussion.

8.2 Classification experiment

This study [68] describes a new, community-based strategy for defining a morphological taxonomy in order to establish a list of terms that could be used by all researchers to distinguish neuronal morphologies. In particular, 42 expert neuroscientists examined 320 interneurons, assigning every neuron to a limited number of neuron types, based on studies performed over the years in many laboratories. The aim of this experiment was to propose a consensus terminology regarding the types of interneurons that could be used as the basis for classification. Hence, every neuron could be labeled as one of the fixed types according to majority vote among experts' assignments.

Although interesting conclusions arose from this classification, final conclusions about types of interneurons were not easily drawn since experts did not agree about most of assignments. Thus, the utility of some commonly used terms for the classification of neurons is rather limited. Nevertheless, some knowledge can be extracted regarding the interneuron labels, depending on the number of votes given to each neuron by experts.

Table 8.1 summarizes the number of neurons that were voted by experts depending on the type of interneuron (1 to 10 or uncharacterized) and the number of obtained votes (threshold). Note that the total number of neurons in the table is 241 instead of 320, since although there were 320 reconstructed neurons, only 241 of these neurons were 3-dimensional reconstructions. This kind of reconstruction is necessary for a correct features extraction process. These data are shown from a threshold of 21 votes, which is half the total number of experts. The number of neurons that would either lose (unlabeled) or maintain (labeled) the labels depending on the threshold.

Threshold	U	1	2	3	4	5	6	7	8	9	10	Unlabeled	Labeled
21	11	10	6	2	27	33	0	20	0	2	0	130	111
22	11	5	5	2	25	26	0	19	0	1	0	147	94
23	10	3	4	2	24	22	0	16	0	1	0	159	82
24	10	1	4	2	24	19	0	13	0	1	0	167	74
25	9	1	4	2	22	12	0	12	0	0	0	179	62
26	9	1	4	2	22	10	0	12	0	0	0	181	60
27	8	1	3	2	20	5	0	10	0	0	0	192	49
28	8	1	3	2	20	3	0	6	0	0	0	198	43
29	6	1	3	1	20	3	0	4	0	0	0	203	38
30	5	0	3	1	19	1	0	3	0	0	0	209	32
31	4	0	3	1	19	0	0	2	0	0	0	212	29
32	3	0	3	1	19	0	0	2	0	0	0	213	28
33	3	0	3	1	17	0	0	1	0	0	0	216	25
34	3	0	3	1	17	0	0	0	0	0	0	217	24
35	2	0	2	1	14	0	0	0	0	0	0	222	19
36	2	0	2	1	13	0	0	0	0	0	0	223	18
37	2	0	2	1	10	0	0	0	0	0	0	226	15
38	2	0	1	1	8	0	0	0	0	0	0	229	12
39	2	0	1	0	5	0	0	0	0	0	0	233	8
40	1	0	0	0	3	0	0	0	0	0	0	237	4
41	0	0	0	0	2	0	0	0	0	0	0	239	2
42	0	0	0	0	0	0	0	0	0	0	0	241	0

Table 8.1: Number of neurons voted for each corresponding type depending on different thresholds (number of experts' coincidences). The list of labels (types of interneurons) is: U = uncharacterized, 1 = `common type`, 2 = `horse-tail`, 3 = `chandelier`, 4 = `Martinotti`, 5 = `common basket`, 6 = `arcade`, 7 = `large basket`, 8 = `Cajal-Retzius`, 9 = `neurogliaform`, 10 = `others`.

The lack of agreement among experts can be seen in Table 8.1. The number of neurons that would be labeled according to a high number of experts (votes) is very limited. For instance, if a threshold of 40 was considered, i.e., 40 experts agreed, there would be only 4 labeled neurons. When the threshold is decreased, the number of labeled neurons slightly grows step by step.

This classification experiment is very useful for the current study in order to obtain a labeled data set: the aim is to reach a trade-off between the number of labeled neurons and the labels reliability. The higher the threshold, the fewer labeled instances but the higher the supposed reliability of these labels. It is necessary to take into account that a minimum number of neurons is also necessary to achieve a reasonable statistical result. The selected threshold for the beginning of this study is 22 (the half of experts plus one). This threshold will be later increased until 26 later trying to obtain more detailed conclusions.

8.3 Experimental results

8.3.1 Data

By using a threshold of 22 experts as commented above, the labeled data are 94 neurons[1]. These neurons were reconstructed from brain slices of the cortex of different animals (mice, rats, cats, rabbits, and monkeys), and even from human beings. Taking into account that 11 out of these 94 neurons were labeled as uncharacterized neurons, that the label `common type` can be considered a hodgepodge with neurons that are not clearly identified, and that `chandelier` and `neurogliaform` are types with only two and one neurons (see Table 8.1), respectively, the labeled data then consisted of 75 neurons divided into: 5 `horse-tail` (HT), 25 `Martinotti` (MT), 26 `common basket` (B), and 19 `large` B. Also, since the results from the classification experiment revealed that the distinction between common and large B was not very clear in the biological community, both common and large B were merely labeled as B. Therefore, this study is performed from now on using three different groups of interneurons.

Finally, after some preliminary results, one neuron, which was labeled as MT by 22 experts, was identified as uncharacterized. Therefore, this neuron was removed from this study. Therefore, the final number of labeled neurons was 74. Figure 8.1 shows an example of 3-dimensional reconstructions of each used neuron type.

Like in the previous chapter, Neurolucida (MicroBrightField) was used to extract around 2900 features for each neuron. This vast amount of features due to dendrites and axon of each neuron are divided into different segments and nodes, and each of these parts are detailed in terms of angles and other different analysis (see Section 7.2.1). A subset of these features was selected by the expert, obtaining a final subset of 224 features to characterize each neuron: 10 features describe the soma, 128 describe the axonal characteristics, and 86 features describe the dendrites. Depending on the used features, different data sets were created. Four data sets were used throughout this study: *axon*, *dendrites*, and *soma*, and *all* when the total number of features were used.

8.3.2 Evaluation process

There are two main steps in this study trying to obtain different conclusions. The aims of each step are detailed below. Each step is performed four times depending on the used data,

[1]Each neuron is identified with a unique number that will be used throughout the chapter

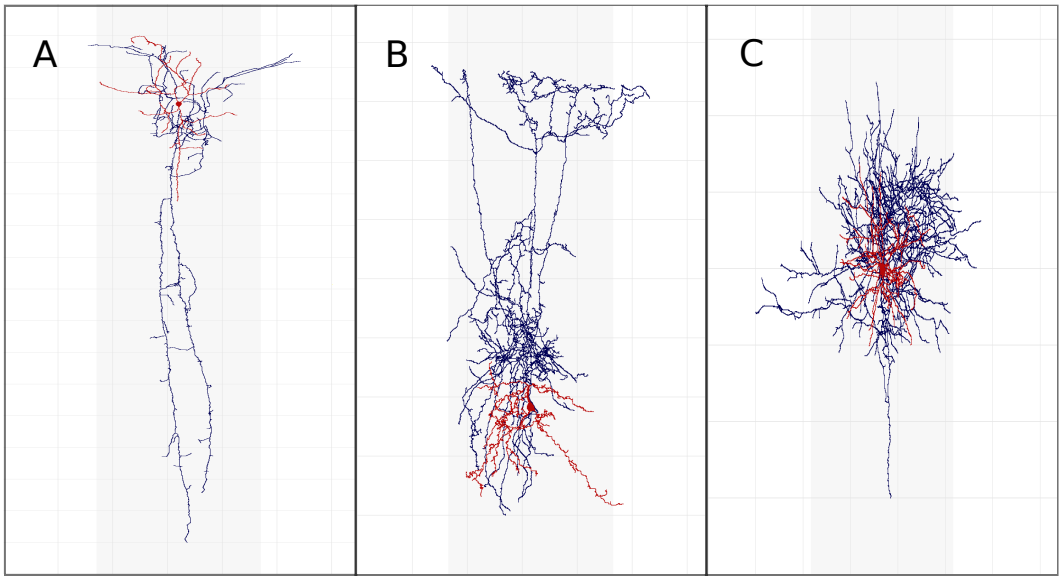


Figure 8.1: 3-dimensional reconstruction of three neurons used in this experiment: (A) neuron 39 labeled as `horse-tail` (HT), (B) neuron 73 labeled as `Martinotti` (MT), and (C) neuron 296 labeled as `basket` (B).

with different features in each case as commented above. The main goal of using different data sets is to draw conclusions about which parts of a neuron are the most important for a classification task and for finding new subtypes of neurons. The steps are:

1. Supervised classification step (see Section 2.2). This step is performed taking advantage of the available labels. The used algorithm is NB (see Section 2.2.1) and validation results are estimated by using 10-fold cross-validation (see Section 2.2.2). The aim of this first step is to check the reliability of labels. Note that the reliability of labels was previously taken into account by selecting a threshold of 22, but it is possible that the morphological features and the used algorithm are not able to correctly classify the neurons. Besides the original data with all the 224 features filtered by the expert, an FSS process is also run for each kind of features to select the most interesting ones according to a wrapper approach using GA as heuristic for searching an optimal subset (see Section 2.5).

2. Semi-supervised clustering step (see Section 2.4.2). The used algorithm is 3SMBC, created in this thesis and detailed in Chapter 5. The aim of this step is to obtain more conclusions about the separation of the known types of interneurons and also about possible subtypes that can be identified for each type. Each experiment in this step consists of completely hiding the labels of instances from one interneuron type, whereas the remaining instances maintain their labels (partially labeled data). The unlabeled instances are then clustered into either one of the known types or into a new subtype. For each data set, three experiments are run (hiding one type of interneuron each time).

8.3.3 Results

Results of each step are presented next. The details of each step and possible relations between the obtained results are also analyzed.

A. Supervised classification

Results for the supervised step are divided depending on the kind of features used and whether the FSS process is run or not.

All data set

- No FSS. The supervised classification results using *all* are shown in Table 8.2. The obtained accuracy was 87.84% with 9 misclassified neurons. These misclassified neurons are the first example about either problems with the reliability of labels or limitations of NB using features from all parts of the neurons. Regarding each specific type, only one HT was correctly classified, whereas the other four were misclassified as MT. Again four MT were misclassified as B and only one neuron with B label was misclassified as MT.

		Predicted class		
		HT	MT	B
Real class	HT	1	4	0
	MT	0	20	4
	B	0	1	44

Table 8.2: Supervised classification results classifying three known types of interneurons using 224 features of all parts of the neurons (*all*) without FSS. The number of correctly classified instances was 65 (87.84%) while the misclassifications were 9 (12.16%).

  The misclassified instances were:

  - HT: 10, 72, 89, 142 (as MT).
  - MT: 53, 127, 184, 185 (as B).
  - B: 84 (as MT).

- FSS. After the wrapper FSS, 111 features out of 224 were selected as the best feature subset. Results using these features are shown in Table 8.3. The accuracy was 90.54%, obtaining a better result than when no FSS was performed in the previous experiment. The improvement was achieved because one HT and two MT, which had been misclassifications before, were correctly classified here. However, one B, which had been correctly classified before, became a misclassification. Regarding the selected features, 71 were axonal, 35 dendritic, and 5 soma features. Thus, the percentages related to the total number of features from each part were: approximately 55% of axon features, 40% of dendritic features, and 50% of soma features. Regarding the total number of

selected features, almost 65% of features were related to the axon, around 31% were dendritic features, and 4% of features contained information about the soma.

		Predicted class		
		HT	MT	B
Real class	HT	2	2	1
	MT	0	22	2
	B	0	2	43

Table 8.3: Supervised classification results classifying three known types of interneurons using features of all parts of the neurons (*all*) with FSS (111 features). The number of correctly classified instances was 67 (90.54%) while the misclassifications were 7 (9.46%).

The misclassified instances were:

- HT: 10, 89 (as MT), 142 (as B).
- MT: 127, 185 (as B).
- B: 84, 131 (as MT).

***Axon* data set**

- No FSS. Supervised classification results using the 128 features containing information about the axon of each neuron are shown in Table 8.4. The accuracy was again 90.54%, the same than when the best identified subset of features from all the parts of the neurons was used. Although the general results were the same and the accuracy estimated using the that axonal features was the same as the obtained using the best feature subset previously obtained from all parts of the neuron, note that there were some differences in the misclassifications in some cases. Thus, the final results were the same in accuracy terms, but different features are able to distinguish better some neurons and worse others.

		Predicted class		
		HT	MT	B
Real class	HT	3	1	1
	MT	0	21	3
	B	0	2	43

Table 8.4: Supervised classification results classifying three known types of interneurons using 128 features regarding the axon (*axon*) without FSS. The number of correctly classified instances was 67 (90.54%) while the misclassifications were 7 (9.46%).

The misclassified instances were:

- HT: 142 (as MT), 10 (as B).
- MT: 127, 184, 185 (as B).
- B: 84, 170 (as MT).

- FSS. After the wrapper FSS on *axon*, 44 features were selected. Supervised classification results using this feature subset are shown in Table 8.5. The obtained accuracy reached 91.89%, which is the highest value obtained in this set of experiments. Thus, a "clean" (after FSS) axon feature subset is the best option to distinguish among HT, MT, and B interneurons. Nevertheless, there were misclassified neurons (four HT and 2 MT) when it might have been expected that, with the chosen threshold, the labels of all the neurons were correct.

		Predicted class		
		HT	MT	B
Real class	HT	1	1	3
	MT	0	22	2
	B	0	0	45

Table 8.5: Supervised classification results classifying three known types of interneurons using features regarding the axon (*axon*) with FSS (44 features). The number of correctly classified instances was 68 (91.89%) while the misclassifications were 6 (8.11%).

The misclassified instances were:

  - HT: 39 (as MT), 72, 89, 142 (as B).
  - MT: 127, 185 (as B).

***Dendrites* data set**

- No FSS. Supervised classification results using the 86 dendritic features are shown in Table 8.6. These results were not as good as the previously obtained using *all* and *axon*. The accuracy decreased until 74.32%. None of the HT neuron was correctly classified, while 6 and 8 MT and B, respectively, were also misclassified.

		Predicted class		
		HT	MT	B
Real class	HT	0	3	2
	MT	0	18	6
	B	0	8	37

Table 8.6: Supervised classification results classifying three known types of interneurons using 86 features regarding the dendrites (*dendrites*) without FSS. The number of correctly classified instances was 55 (74.32%) while the misclassifications were 19 (25.68%).

The misclassified instances were:

  - HT: 10, 72, 142 (as MT), 39, 89 (as B).
  - MT: 34, 53, 127, 222, 249 (as B).
  - B: 21, 93, 131, 176, 230, 286, 295, 296 (as MT).

- FSS. The FSS process selected 35 dendritic features. Results using this feature subset are shown in Table 8.7. The accuracy was 72.97%, which is lower than the accuracy obtained without FSS. This is the only experiment in which the obtained result was lower with FSS than without it. In any case, results were very similar to the obtained using *dendrites* without FSS and the conclusion for both cases is that dendritic features are not as useful as axonal features for distinguishing the three types of interneurons with NB.

		Predicted class		
		HT	MT	B
Real class	HT	0	4	1
	MT	1	16	7
	B	0	7	38

Table 8.7: Supervised classification results classifying three known types of interneurons using features regarding the dendrites (*dendrites*) with FSS (35 features). The number of correctly classified instances was 54 (72.97%) while the misclassifications were 20 (27.03%).

The misclassified instances were:

- HT: 10, 72, 89, 142 (as MT), 39 (as B).
- MT: 107 (as HT), 34, 53, 127, 184, 222, 249, 291 (as B).
- B: 21, 93, 131, 176, 286, 295, 296 (as MT).

Soma data set

- No FSS. The soma is a part that has small differences among neurons. It can be seen in the results shown in Table 8.8. The accuracy was only 52.7%. The small differences in the soma among neurons were also reflected in that many neurons were classified as B. This happened because the B group was the biggest group, and NB classifies instances into the biggest group when there are no many differences among the features.

		Predicted class		
		HT	MT	B
Real class	HT	2	0	3
	MT	0	5	19
	B	5	8	32

Table 8.8: Supervised classification results classifying three known types of interneurons using 10 features regarding the soma (*soma*) without FSS. The number of correctly classified instances was 39 (52.7%) while the misclassifications were 35 (47.3%).

The misclassified instances were:

- HT: 10, 39, 142 (as B).
- MT: 3, 20, 34, 53, 66, 69, 77, 82, 106, 107, 111, 127, 178, 184, 222, 231, 249, 291, 302 (as B).

  - B: 9, 44, 76, 120, 296 (as HT), 21, 93, 103, 166, 170, 176, 216, 238, (as MT).

- FSS. Only two features were selected by the FSS process. Results (Table 8.9) only improved very slightly after FSS, obtaining 54.05% of accuracy, which is clearly not enough to distinguish these types of interneurons. Thus, soma features are not as useful as axonal features, or even as dendritic features, to classify the types of interneurons with NB.

		Predicted class		
		HT	MT	B
Real class	HT	0	0	5
	MT	0	5	19
	B	5	5	35

Table 8.9: Supervised classification results classifying three known types of interneurons using features regarding the soma (*soma*) with FSS (2 features). The number of correctly classified instances was 40 (54.05%) while the misclassifications were 34 (45.95%).

The misclassified instances were:

  - HT: 10, 39, 72, 89, 142 (as B).
  - MT: 3, 20, 53, 66, 69, 77, 82, 106, 111, 127, 178, 184, 197, 222, 231, 249, 291, 302 (as B).
  - B: 44, 76, 120, 131, 225 (as HT), 5, 55, 170, 216, 238 (as MT).

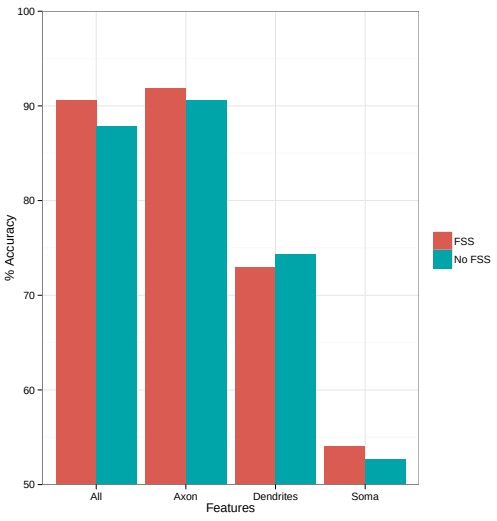


Figure 8.2: Summary of accuracy values obtained with NB depending on the used data.

A summary of all the obtained accuracy values is shown in Figure 8.2. It can be seen how using *soma* or *dendrites* separately is not enough to achieve a good separation of HT, MT, and B neurons. When these subsets are used together with the axon subset (*all*), results increased until more than 90% of accuracy after FSS. Nevertheless, according to these results, it is not necessary to use all the subsets since the best results were achieved by using only *axon* after FSS. The FSS process obtained slightly better results than when no FSS was carried out for all cases except for *dendrites*.

Detailing the misclassifications, HT neurons were not well-separated from the other groups, in spite of this type of neurons is very different from the other ones, as can be seen in Figure 8.1. This fact might happen

due to the unbalanced number of available neurons, since the number of HT neurons is very low compared to the number of MT and B neurons (see Table 8.1). The semi-supervised approach overcomes this problem (see the next section). Other misclassifications that were repeated several times, even when the best results were achieved, are the neurons numbered as 127 and 185, labeled as MT, and the neuron 84, labeled as B. These neurons are shown in Figure 8.3.

Although these neurons have clear differences with respect to typical neurons presented in Figure 8.1, visual expert validations confirmed that they were correctly labeled. Thus, this is a very representative example of lack of agreement in the different validations, since the machine learning and the expert validations did not agree. These neurons should be considered as misclassifications according to the best results of NB, and the opinion of the expert is that their labels are correct. Note that if this study is presented to another expert, validation may be different.

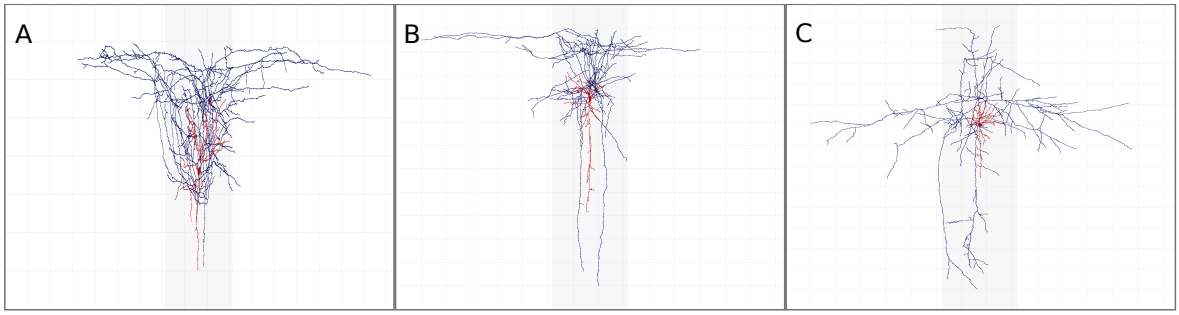


Figure 8.3: 2-dimensional reconstructions of three misclassified neurons in most supervised scenarios. (A) neuron 127, `Martinotti` (MT), (B) neuron 185, `Martinotti` (MT), and (C) neuron 84, `basket` (B).

B. Semi-supervised clustering

The process to achieve the partially labeled data that 3SMBC needs as input data is based on hiding the labels of instances that belong to a specific type of interneuron. The remaining instances (labeled as one of the other two types of interneurons) maintain the label as known information for the algorithm. This process is repeated for each data set. By using this process, some conclusions can be drawn about whether the different feature subsets are able to correctly separate each type from the others, and also whether new subtypes are identified for each type of interneuron. At the end of the algorithm, the unlabeled instances may belong either to a new cluster or to some of the two known groups. Both the known and the new clusters are identified by weighting the features according to their relevance provided by the algorithm (see Section 5.3).

Results are separated depending on the hidden interneuron type. Note that only results of unlabeled instances are shown because the labeled instances do not change their labels at the end of 3SMBC. As can be seen in more detail in each subsection and as expected according to previous experiments, the results were different depending on the used data set. *Soma*

features separately are not shown in this experiment because they do not contain enough information to obtain remarkable results, as shown in the supervised classification step, and as confirmed in (not shown) preliminary results with the semi-supervised subspace clustering algorithm.

Hiding HT

The instance assignments of the five hidden HT neurons are shown in Table 8.10. Each number (in this case only 1) corresponds to a new subgroup, whereas MT and B correspond to instances that were grouped into those known groups (misclassifications according to the available labels). The separation of HT neurons from MT and B groups was better than that obtained in the supervised classification step. This separation was complete when *all* and *axon* data sets were used.

	Data		
ID	*all*	*axon*	*dendrites*
10	1	1	1
39	1	1	1
72	1	1	MT
89	1	1	1
142	1	1	1

Table 8.10: Semi-supervised assignments when labels of `horse-tail` (HT) instances were hidden taking into account a threshold of 22 for each of the considered data sets.

Details for each considered data set (depending on the used features) are commented next:

- Using *all* data set. A new subgroup was found for the five neurons. This result is important because a complete separation of HT with respect to MT and B was not achieved in the supervised classification step. Therefore, it can be concluded that HT is correctly distinguished from the other two types of interneurons when each type is characterized with a different subspace. The number of most relevant[2] features was 75, which is around 33% of the total features. Regarding each part of the neuron, 72% of the 75 features were from the axon, 25% from dendrites, and 3% from soma.

- Using *axon* data set. Again, only one subgroup was found for the five HT neurons. The number of relevant features was 53. All the selected features but one completely matched with the axonal features previously identified as relevant using *all*. Besides, 98% of the features matched up with the previous subset. Therefore, it can be concluded that a subset of axonal features was enough to distinguish the available HT neurons from MT and B. Specifically, some of the most important features were: all the sholl axon features regarding distances from 120 µm to 420 µm, the neuron orientation

[2]A feature is considered as relevant for a subspace if 3SMBC assigns a weight (ρ, see Equation 5.31) of, at least, 0.8.

(with many polar axon features), and the information about the local branch angle of the axon.

- Using *dendrites* data set. There was a new subgroup for four HT neurons, but neuron 72 was clustered into the MT group. This neuron was also misclassified as MT in the supervised classification step using the dendritic features.

Hiding MT

The instance assignments of the 24 hidden MT neurons are shown in Table 8.11. Each number in the table corresponds to a new subgroup (in this case 1, 2, 3, and 4, depending on the used data), whereas HT and B correspond to instances that were grouped into those known groups (misclassifications according to the available labels). Details for each considered data

	Data		
ID	*all*	*axon*	*dendrites*
3	2	2	3
20	1	1	1
34	B	B	1
53	B	B	4
66	1	3	4
69	2	B	2
73	2	B	1
77	2	2	1
82	2	2	1
106	1	1	1
107	B	B	HT
111	B	B	1
127	B	B	B
178	2	2	1
184	2	2	B
185	B	B	4
222	2	2	HT
229	1	2	4
231	3	2	2
249	1	1	2
272	3	2	3
290	B	B	2
291	2	2	4
302	2	2	2

Table 8.11: Semi-supervised assignments when labels of `Martinotti` (MT) instances were hidden taking into account a threshold of 22 for each of the considered data sets.

set (depending on the used features) are commented next:

- Using *all* features. Three subgroups were found and seven neurons were misclassified and grouped into the B cluster. The misclassified neurons were: 34, 53, 107, 111, 127, 185, and 290. The number of misclassified neurons is higher than that obtained in the

supervised classification approach, where four and two MT neurons were misclassified by using *all* without and with FSS, respectively. Neurons 127 and 185 are within the misclassification subset. These two neurons were misclassifications using the NB algorithm in the previous step when both *all* and *axon* data sets were used.

- Using *axon* features. The algorithm also found three subgroups, but one of them has only neuron 66. The number of misclassified neurons was even higher than using *all*. In total, nine neurons were misclassified and grouped into the B cluster. Again, neurons 127 and 185 were misclassified neurons.

- Using *dendrites* features. The information given by dendrites separately is completely different. Four subgroups with four misclassified neurons (two of them joined to the HT cluster) were found. The number of misclassified neurons using the NB algorithm and the *dendrites* data were 18 and 17, with no FSS and with FSS, respectively. Therefore, 3SMBC obtained a more accurate results with only four misclassifications.

Hiding B

The instance assignments of the 45 hidden B neurons are shown in Table 8.12. Each number corresponds to a new subgroup (in this case 1, 2, 3, and 4, depending on the used data), whereas HT and MT correspond to instances that were grouped into those known groups (misclassifications according to the available labels).

Details for each considered data set (depending on the used features) are commented next:

- Using *all* features. Three new subgroups were found with one misclassified neuron (9). One of these subgroups had the 80% of the B neurons, while the explanation for the other two subgroups was not clear for the expert.

- Using *axon* features. There were two identified subgroups with neuron 84 as a misclassification. Neuron 84 was identified, also considering supervised classification results, as a difficult target for this kind of algorithms. In this case, the percentage of B neurons grouped into the biggest group is almost 90%.

- Using *dendrites* features. In this case, there were not misclassifications. Therefore, and similarly to results by hiding MT neurons, the number of misclassified neurons was again lower by using *dendrites* than *all* or *axon*. There were four subgroups, and again, one of them has most of the neurons with almost 80% of the total number of B neurons.

Some preliminary conclusions could be drawn from these results:

- Regarding HT neurons, the obtained separation was complete and only one subgroup was found. This is an interesting advance with respect to the supervised classification results, since these neurons were not completely distinguished from MT and B neurons

	Data		
ID	*all*	*axon*	*dendrites*
5	1	2	2
9	HT	2	2
11	3	2	2
12	1	1	2
21	3	2	2
26	3	2	3
29	3	2	2
35	3	2	2
44	3	2	2
55	3	2	2
61	3	2	3
63	2	1	2
76	3	2	3
78	3	2	1
81	3	2	2
84	2	MT	2
93	1	1	1
96	3	2	2
103	1	1	3
108	3	2	1
109	3	2	4
113	3	2	2
120	3	2	2
121	3	2	2
131	3	2	2
143	3	2	2
146	3	2	2
161	3	2	2
166	3	2	2
170	2	2	2
176	3	2	1
202	3	2	2
214	3	2	2
216	1	2	4
225	3	2	2
230	3	2	2
236	3	2	2
238	3	2	2
240	3	2	2
247	3	2	2
283	3	2	2
286	3	2	1
293	3	2	3
295	3	2	2
296	3	2	2

Table 8.12: Semi-supervised assignments when labels of `basket` (B) instances were hidden taking into account a threshold of 22 for each of the considered data sets.

by using the NB algorithm. Therefore, detailing each found cluster with a specific subspace supports the separation of HT neurons from MT and B neurons. Specifically, the most important features for the HT subspace were related to the axon (sholl axon and polar axon features).

- Regarding MT neurons, the obtained separation was not complete. There were several misclassified neurons and, according to these results, the dendritic information became more important for this separation. This is interesting because the most important part of the neuron to obtain the previous HT separations was the axon. In any case, the complexity of these neurons is higher than HT neurons since although at least 22 experts labeled these neurons as MT, neither a supervised classification nor a semi-supervised approach were able to obtain a complete separation.

- Regarding B neurons, the separation was almost complete by using *all* and *axon*, and complete by using *dendrites*. Again, this indicates the possible relevance of dendritic features. This possible relevance was not revealed by using NB. There was a big subgroup of B (numbered as 2 and as 1 in Table 8.12 for *all* and *axon*, respectively) that could be considered as a good starting point of "basic B neurons", while some other neurons (like neuron 84) were more problematic.

The conclusions for HT neurons can be considered as remarkable. However, the expert's validation for the identified subgroups for MT and B neurons was not as satisfactory. For this reason, the next step was to reduce the data set due to the complexity of some of the current MT and B neurons. This reduction can be obtained by increasing the threshold commented in Section 8.1. The main new goal is to obtain more specific conclusions about MT and B neurons using a data set with more reliable labels.

Different threshold

The new selected threshold was raised to 26, i.e., at least 26 experts assigned to each neuron the same label. According to this threshold, there are 22 neurons labeled as MT and 22 neurons labeled as B (10 common B and 12 large B), as can be seen in Table 8.1. Therefore, there are 44 neurons for each new data set. Four data sets are again considered because the same separation regarding the features is used.

Neurons 127 and 185, which were misclassified neurons for the best supervised classification results and also for the semi-supervised approach, were not included in this new set of data. Nevertheless, neuron 84, which was also problematic, was maintained since it was labeled as B by 33 experts.

The same semi-supervised approach and process that was previously used was again employed for this new data set. In this case, there are only two different labels and, therefore, MT were first hidden maintaining B labels and with the aim of obtaining a good separation of MT from B neurons and discovering possible MT subgroups. This process was then repeated, but hiding the B labels and maintaining MT labels.

Before showing the results obtained with the semi-supervised approach, Table 8.13 summarizes the results obtained using the same supervised approach than before, i.e., using NB with no FSS and with wrapper FSS. It can be seen how results were similar to the obtained with the previous threshold, even when the number of neurons was decreased and only two labels were used. The obtained results with the threshold 26 were similar to the obtained with the threshold 22 in terms of both accuracy and importance of features, since again axonal features were more accurate at distinguishing between the two types of interneurons than when all features were used. As a curiosity, the FSS process did not achieve better results than when the original data were used for the two highest accuracy values (using *all* and *axon*). It may reveal that the maximum accuracy value was obtained in each case and, although the FSS process obtained models with a lower number of features, the obtained accuracy did not improve. It is interesting the similarity of these results with the previously obtained, but this work focuses on the semi-supervised approach and its results, which are presented next.

	Data			
	all	*axon*	*dendrites*	*soma*
No FSS	88.63%	90.91%	70.45%	47.72%
FSS	88.63%	90.91%	72.72%	61.36%

Table 8.13: Accuracy obtained using NB as supervised approach (estimated with a 10-fold cross validation) depending on the data used and whether FSS was carried out.

Hiding MT

The instance assignments are shown in Table 8.14. Each number corresponds to a new subgroup (in this case 1, 2, 3, and 4, depending on the used data), whereas B corresponds to instances that were grouped into that known group (misclassifications according to the available labels). The main conclusion from assignments presented in Table 8.14 is that the number of misclassifications was increased when only *axon* or *dendrites* were used. On the contrary, when all parts of the neuron were combined, there was only one misclassification: neuron 53 was grouped into the B group. Note that neuron 53 was also misclassified when NB was applied in the supervised approach in all scenarios except for the *axon* data. Thus, the used algorithms, together with the available features, were not able to clearly consider this neuron as a typical MT neuron.

After expert's validation, the neuroscientist was able to visually distinguish some different subtypes of MT. According to these and other preliminary results, MT neurons seem a population of neurons more heterogeneous than previously thought.

Hiding B

The instance assignments are shown in Table 8.15. Each number corresponds to a new subgroup (in this case 1, 2, and 3, depending on the used data), whereas MT corresponds

	Data		
ID	*all*	*axon*	*dendrites*
3	3	4	2
20	1	1	2
34	3	B	1
53	B	B	2
66	1	1	1
69	4	B	1
73	3	B	1
77	2	3	2
82	3	4	4
106	1	1	3
107	2	B	1
111	2	B	1
178	2	3	1
184	4	3	B
185	3	3	2
222	3	2	2
229	1	2	4
231	3	2	B
249	1	1	4
272	3	2	3
291	3	2	B
302	3	2	4

Table 8.14: Semi-supervised assignments when labels of `Martinotti` (MT) instances were hidden taking into account a threshold of 26 for each of the considered data sets.

to instances that were grouped into that known group (misclassifications according to the available labels).

Results in Table 8.15 were the most interesting results for our expert. Neuron 84 was the single misclassified neuron when *all* and *axon* were used. This neuron was previously commented because it was very problematic for NB during the supervised classification step. Results confirmed again that either this neuron might not be correctly reconstructed or the available features were not able to characterize it.

Traditionally, axonal features have been considered by neuroscientist as the most important characteristics to classify neurons. However, as can be seen in Table 8.15, no subgroups were identified when *all* and *axon* data sets were used. Thus, these results indicate axonal features allow for the successful separation of B neurons from MT neurons, whereas no separation among B neurons was observed.

However, when *dendrites* data were used, the algorithm found three subgroups. After expert validation, these subgroups can be considered as logical groups, according to the results expected by the expert, depending on both the size and the orientation of the neurons. One representative neuron from each of the three found subgroups is shown in Figure 8.4. Neurons from the first subgroup (Figure 8.4 (A)) are big neurons that mainly match with the previously group of large B neurons (see Table 8.1). These neurons have also an elongated axon arborization. Neurons from the second subgroup (Figure 8.4 (B)) are also considered

	Data		
ID	*all*	*axon*	*dendrites*
5	1	1	3
9	1	1	1
11	1	1	1
12	1	1	1
21	1	1	3
26	1	1	2
55	1	1	1
76	1	1	1
78	1	1	2
84	MT	MT	1
93	1	1	2
113	1	1	3
131	1	1	1
214	1	1	1
216	1	1	2
225	1	1	1
230	1	1	1
247	1	1	3
286	1	1	1
293	1	1	1
295	1	1	3
296	1	1	3

Table 8.15: Semi-supervised assignments when labels of `basket` (B) instances were hidden taking into account a threshold of 26 for each of the considered data sets.

as large B neurons, but the arborizations are horizontally distributed. Figure 8.4 (B) was rescaled to maintain the same size than the other two neurons, as can be seen in the elongated squares that indicate the size of the neuron. Finally, the third subgroup contains neurons like Figure 8.4 (C), which are smaller than neurons from the first and second subgroups, and mainly match with the previously assigned name common B neurons.

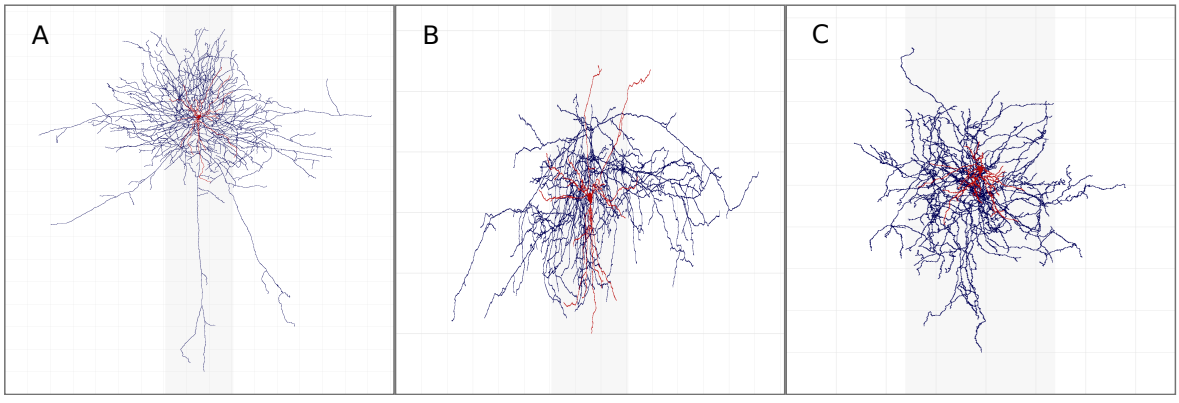


Figure 8.4: 2-dimensional reconstructions of three types of `basket` (B) neurons. (A) Neuron 12 belongs to the first subgroup, (B) neuron 78 belongs to the second subgroup, and (C) neuron 21 belongs to the third subgroup. All these subgroups were found using dendritic features.

The most interesting conclusion about the identified subgroups is that dendritic features were able to identify neurons containing particular axonal arborization. Therefore, it is possible that, according to the obtained results with the available neurons, dendritic and axon arborizations may be more related than previously thought.

8.4 Summary and discussion

After an interesting experiment that was performed showing more than 300 interneurons to many experts, there were three types of interneurons that might be considered as known types: `horse-tail` (HT), `Martinotti` (MT), and `basket` (B). The reliability of the assigned labels depended on the number of experts that voted the same label for a neuron. It can be seen from those results the lack of agreement that exists. When a supervised classification algorithm (naïve Bayes (NB)) was applied trying to distinguish these three types, the main conclusion is that some neurons were not correctly classified. This happened mainly for HT neurons. Movements toward a final classification of neurons are not easy, since not even three very known types of interneurons, using a supposedly reliable data set, can be automatically distinguished with NB. For this task, axonal features were identified as the most important features to distinguish the types of interneurons, since when axonal features were combined with dendritic and soma features, the classification accuracy decreased. The importance of the axonal features was expected by an expert neuroscientist used as external validation.

It is necessary to take into account that the first problem at classifying neuronal data begins in the reconstruction step. All the available neurons are not complete neurons, thus the reconstructed morphologies may be or may not be very representative data about the exact neurons. Another very important milestone that should be reached to further advances is related to the morphological features measured. Throughout this study, a complete list of 224 features from each part of the neuron was used to characterize them. Nevertheless, new complementary parameters should be achieved in order to obtain a complete description of the morphology of the neuron.

3SMBC was then applied as semi-supervised approach. The labels of each type of interneurons were hidden in each case while the other types maintained these labels, obtaining partially labeled data. The goal of this approach was to obtain not only a good separation among the types of interneurons, but also new subgroups for each type. The results throw some light about the proposed types of interneurons and their possible subtypes. HT neurons were correctly separated by using this semi-supervised approach when the threshold of 22 was used. This good separation was not achieved with NB. However, the task became more challenging for MT and B neurons, because there were misclassified neurons and the subgroups were not very clear after expert's validation.

When the threshold was then raised until 26, obtaining more reliable data labels, some interesting conclusions arose after repeating the study with the new data, which contained only MT and B neurons. Both MT and B types of neurons were completely separated from the other group except for one neuron using all features. This is interesting because conclusions

about the features from supervised classification results were that axonal features are the most important for distinguishing different types of interneurons, and that the combination of axonal features with features from other parts of the neuron led to worse results.

MT neurons were separated into different subgroups. This happened throughout all preliminary and final experiments. The MT neurons are considered as a very basic neuron and subgroups were not previously identified. However, a final specification of these groups was not achieved because, as the expert expected at the beginning of the experiment, new features should be extracted from the neuron reconstructions in order to specify some relationships between the different parts of each neuron.

Finally, the most remarkable conclusion is related to B neurons. Three B subgroups were identified, matching with some previous expert intuition: a subgroup with large and elongated neurons, another one with large and horizontally distributed neurons, and, finally, a subgroup with smaller B neurons. The most interesting conclusion about these subgroups is that dendritic features were used to identify them, while the main characteristics of these neurons are related to the axon. This fact does not mean that the axon is not a relevant feature when characterizing a neuron, but instead reveals that dendritic characteristics in neurons could be more related to axonal characteristics than previously thought. This could be a very interesting finding that deserves further studies to be confirmed.

Part V

CONCLUSIONS AND FUTURE WORK

Chapter 9

Conclusions

The main and specific conclusions drawn from this thesis have been presented throughout each chapter. The most relevant conclusions will be summarized in this chapter, emphasizing the reached achievements.

- Although clustering validation is still an open field of research, some internal indices can be used to guide the clustering solution. The performance of these measures is very dependent on several factors, e.g., used clustering algorithm or data characteristics. According to the obtained results, some measures such as Calinski, Silhouette, and Gamma, should be chosen before others such as C-index and DB, almost systematically. Nevertheless, the final selection of an index should be made depending on some previous analysis on data.

- Data labels can be used together with searching for subspaces to enhance clustering results when dealing with data with structures hidden in different subspaces. Two different algorithms have been created obtaining this main conclusion. Some more specific conclusions are:

  - Available data labels can be used not only to validate, but also to guide the created algorithms. Besides, data labels can be used from different perspectives as can be seen in the algorithms. On the one hand data labels were mainly used to identify the most relevant subspaces, while on the other hand they were used as fixed information known beforehand that is introduced into the clustering process to support both the search for subspaces and the instance assignments.
  - Searching for subspaces leads to clustering results that are very accurate when dealing with data with structures hidden in different subspaces. Traditional clustering algorithms may find not such accurate solutions (which may be even unreachable) for this kind of data.

- Traditional and novel machine learning techniques may help advance towards the solution of real and challenging problems. Specifically, the neuronal data classification

is an example of this kind of problems. Some advances have been achieved by using traditional machine learning tasks, and also new algorithms, such as the presented in this work. These advances may help to the understanding of a possible classification of neurons. Nevertheless, more efforts should be made to obtain more conclusive results for this interdisciplinary problem.

The main hypothesis of this thesis: **taking advantage of available data labels, and using different subspaces for finding hidden structures, lead to clustering solutions that, otherwise, cannot be such accurate or even unreachable** was demonstrated by obtaining satisfactory results on the main objective: to tackle the semi-supervised subspace clustering problem using different approaches and to develop the adequate scientific experiments in order to validate the different approaches.

The separated objectives defined in Chapter 1 have been fulfilled as presented during the different chapters of this document:

1. Chapter 3 deals with the objective "Study of some of the most used clustering validation indices under different data conditions and clustering algorithms, as an inherent problem in all tasks related to clustering".

2. The objective "Creation of semi-supervised subspace clustering proposals from two different points of view" is presented in Chapter 4 for the approach based on "Using available data labels for searching for subspaces firstly, before searching for clusters. This proposal assigns each instance to only one cluster (hard clustering) and is based on mapping known labels to subspaces using supervised classification techniques. Subspaces are then used to find clusters using traditional clustering techniques", and in Chapter 5 for the approach based on "Using available data labels to search for subspaces and clusters at the same time in an iterative process. This proposal assigns each instance to each cluster based on a membership probability (soft clustering) and is based on integrating known labels and the search for subspaces into a model-based clustering approach".

3. Chapter 6 presents an introduction to neuroscience as a starting point for Chapters 7 and 8 where the objective "Application of machine learning to neuroscience, presenting a real problem and applying the necessary techniques, including some of the proposals, to provide solutions" is fulfilled.

Besides the chapters related to the commented objectives, Chapter 2 presents a complete overview of all the machine learning topics used throughout this research, presenting different approaches, algorithms, and references.

9.1 Publications

To conclude this chapter, the publications derived from this research are listed below.

- L. Guerra, L. McGarry, V. Robles, C. Bielza, P. Larrañaga and R. Yuste. Comparison between supervised and unsupervised classifications of neuronal cell types: A case study. *Developmental Neurobiology*, 71(1):71–82, 2011. Current JCR (2011): 3.551.

- L. Guerra, V. Robles, C. Bielza and P. Larrañaga. A comparison of clustering quality indices using outliers and noise. *Intelligent Data Analysis*, 16:703–715, 2012. Current JCR (2011): 0.448.

- L. Guerra, C. Bielza, V. Robles and P. Larrañaga. Semi-supervised projected model-based clustering. *Data Mining and Knowledge Discovery*, Submitted. Current JCR (2011): 1.545.

- L. Guerra, V. Robles, C. Bielza and P. Larrañaga. Partially labeled data clustering in subspaces. *International Journal on Artificial Intelligence Tools*, Submitted. Current JCR (2011): 0.217.

Chapter 10

Future Work

The work presented in this thesis mainly addresses semi-supervised clustering, also dealing with completely supervised and unsupervised techniques in different situations together with dimensionality reduction under several approaches. All these topics together compose a vast amount of research and knowledge in the literature. Therefore, and although the objectives of this thesis were satisfactory fulfilled, a series of future works could be considered.

In this chapter some of the most relevant future lines and open issues will be enumerated. Some of them deal with the improvement of the presented proposals, whereas others are related to changes in experiments and applications. The order of these future lines follows the same order as that used during the manuscript.

10.1 Clustering validation

Taking into account that clustering is a descriptive task, and that different descriptions could be equally accepted by different users, validating clustering is a task more based on expert opinions that on numerical measures. Nevertheless, depending on the purpose, some measures can be used to guide the search of subsets of features or number of clusters as seen in Chapter 3. A more extended battery of measures and data characteristics could be studied, trying to drawn some conclusions about indices' behaviour and performance when validating clustering solutions obtained with different algorithms.

Another remark about clustering validation, and maybe as a no realistic desire, the creation of a validation framework capable of validating any clustering solutions would be the ideal aim in this research field. The creation of this framework could be based on the definition of new validation indices, but, as previously commented, due to differences in clustering algorithms, and mainly, due to differences in expert validation opinions, this aim seems far from being reachable. Finally, this desire can be extended to semi-supervised clustering. The clustering validation indices may not be useful to correctly validate semi-supervised approaches. Therefore, new indices to validate this type of problems could be also defined.

10.2 Semi-supervised subspace clustering

Contrary to clustering validation, semi-supervised subspace clustering is open to minor and major future work. This specific pattern recognition task can be considered as recent. It means that improvements for created algorithms and new techniques can be elaborated to solve this kind of problem. Focusing on the two proposals presented in this work, the open lines for each one of them are listed next.

Hard approach

Although some positive conclusions (commented in Section 4.5) are drawn from the proposal called KMF, which was presented in Chapter 4, there are many opportunities for improvements in the proposal. Related to the general ideas in which the created framework is based on, there are three important aspects that can be modified. First of all, KMF is a projected clustering approach since overlapping in clusters is not allowed and each instance belongs to only one cluster in the final solution. To achieve this, a post-process filter is applied to the found clusters to delete possible repetitions of the same instance in different clusters. If this step was withdrawn, the same instance could belong to more than one cluster in the same solution. With this new situation, and according to Section 2.5.1, KMF would become a subspace clustering approach in which overlapped clusters can be found. Both approaches are equally correct depending on the application, and although projected clustering approaches are more easily interpreted, the ad-hoc post-process filter can be seen as a computationally demanding and unnatural step.

The second aspect is related to those instances that are not clustered into any of the genuine clusters, which are found for each known class. According to the current framework, only one subspace is found to cluster all those instances. Using only one subspace may prevent the algorithm from finding hidden structures. Therefore, a new method for finding more than one subspace for those instances that are not clustered into the genuine clusters is desirable. The subspaces associated to the known classes are identified according to the known labels, by using a supervised approach. However, the search for the new subspace during the second phase is based on the results obtained at the end of the first phase and the supervision is rather limited. A possible line to tackle this problem is to introduce an iterative search for subspaces during the second phase. Thus, the first new subspace would be identified as presented in the current proposal and used to cluster the data. Nevertheless, instead of assuming the clustering solution as the correct partition that is included into the final solution, the found clusters would be evaluated including into the final solution only the best cluster(s) according to some validation measure. The instances that still remain unclustered would be then used to identify a new subspace, and the process would be repeated until all instances belong to good clusters, i.e., clusters that obtain a good value according to a validation measure. Again, clustering validation would become a very important step and different solutions would be obtained depending on the validation measure.

Finally, another general aspect that could be modified is related to the identification of

each subspace and the available data labels at the same time. In the current approach, subspaces are identified by using an FSS approach for a binary supervised classification problem without taking into account the hypothetical validation result of each supervised problem. This value may indicate two key aspects: whether the identified subspace is noteworthy and whether the available data labels are reliable. Therefore, according to these results, some flexibility could be introduced into the identified subspaces, allowing either the inclusion or the removal of some features. Besides, if some instances are misclassified in this supervised classification problems, it is possible that labels for those instances are not clear and should be taken wisely to find the genuine clusters. In the current approach, those labels could be removed from the constraint set for the MPCKM algorithm.

Besides the general ideas that could be modified, KMF is a framework that must be instantiated and, therefore, many future work can be made in this direction. Currently, KMF is instantiated by using traditional and available algorithms. However, the instantiation may differ depending on the user and the application domain. The main changes that can be done related to the instantiation are:

- It is necessary to solve an FSS step in a binary supervised classification problem during the search for subspaces in the first phase. A supervised classification algorithm, an FSS approach, and a method to estimate some accuracy measure must be chosen. This is repeated at the beginning of the second phase.

- Once the subspaces are found, a clustering algorithm is used to find the genuine clusters. A clustering algorithm, together with a method to estimate the number of clusters in each solution, must be chosen. Another clustering algorithm is used as final step at the end of the second phase.

As a summary, the decisions about the instantiation must be made regarding how subspaces are identified and how data are clustered using those subspaces. The justification for using some algorithms or others is not trivial because the performance of each algorithm depends on the data. Therefore, the future work related to this should be directed towards experimental work.

Soft approach

Although 3SMBC, presented in Chapter 5, is a closed algorithm that obtained very accurate results in different experimental scenarios, some future work could be done trying to improve, even more, these results. Although others could be considered, three proposal are presented here for further study.

The first proposal is related to the available data labels. It is assumed for 3SMBC, equally than for many other semi-supervised clustering algorithms (it was also commented for KMF), that available data labels are completely reliable. However, if this assumption is not realistic, and the clustering solution is guided by wrong data labels, the output will be also wrong. This possible trouble is very highlighted in the 3SMBC case, since data labels are important

not only taking part in the initialization, but also because they do not change along the complete execution. Thus, it is not allowed that an instance, if was originally mislabeled, can change its label in 3SMBC (unlike in KMF). The first proposal can be tackled from two different points of views. The first one is very simple and consists of allowing the possible changes in data labels for the originally labeled instances. 3SMBC would calculate the new labels for those instances in the same way as is calculated for the unlabeled instances. The second point of view is more complicated and is based on assigning different weights to the instances depending on the reliability of each data label. The subspaces should be generated according to these weights and then, instances that belong to a component according to the original label would be more important to identify the subspace for that component if they have reliable labels than if they have labels with lower reliability.

The second proposal is related to how new components are found and introduced into the 3SMBC mixture model. Currently, a new component is introduced into the model at each level, trying to discover new and hidden data structures. This iterative process is based on initializing the new component according to those instances that worse fit the current components in the model. The main limitation of this approach is that only one component is then considered to be introduced into the model at each level. A new proposal could be based on considering more than one component at each level, based on different initializations. With this proposal, one or more than one new components could be selected to be introduced into the new model.

Finally, the third proposal is related to outliers. Some instances could not properly fit in all the identified subspaces and components. Therefore, they can be considered outliers. Outlier detection is another important paradigm in machine learning, and it is many times related to clustering. An outlier detection method could be introduced into the algorithm under the next definition of outlier: an instance is considered as an outlier if it belongs to a component characterized by using all the available features instead of belonging to some cluster characterized by a subspace. According to it, another component could be considered in the model at each level, characterized by all the available features. If some instances are clustered into that component at the end of the algorithm, they are considered as outliers. Other definitions of outliers may be created, obtaining different outlier detection methods.

10.3 Real applications in neuroscience

Although some conclusions were drawn from the real applications presented in Sections 7 and 8, the final solution for the neuron classification is still remote. Thus, much future work can be considered:

- A better feature extraction process should be designed, trying to gather as much information as possible about neurons in computer features so that these features can be used to obtain better classifications. A possible approach could be related to the extraction of features that gather information about relationships between different parts of the neurons.

- The lack of agreement among experts may be exploited. Some kind of uncertainty (in terms of reliability of labels as votes received) can be introduced into the experiments and new used algorithms to model the agreement.
- More experiments are necessary to obtain final conclusions about relationships between the different parts of neurons.
- Different algorithmic approaches could be applied to the neuron classifications since, after these current experiments, the same neuron may be grouped with different neurons depending on the used data. This is considered in overlapping clustering algorithms and many of them are also related to subspace clustering.
- Finally, and depending on hypothetical results that might be obtained with the previously indicated future work, some kind of consensus approach could be introduced into subspace and semi-supervised clustering approaches if a final solution, with each neuron grouped into only one cluster, is aimed.

Bibliography

[1] E. Achtert, C. Böhm, H. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Detection and visualization of subspace cluster hierarchies. In *Advances in Databases: Concepts, Systems and Applications*, volume 4443 of *Lecture Notes in Computer Science*, pages 152–163. Springer Berlin Heidelberg, 2007.

[2] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J.S. Park. Fast algorithms for projected clustering. *SIGMOD Record*, 28(2):61–72, 1999.

[3] C.C. Aggarwal and P.S. Yu. Finding generalized projected clusters in high dimensional spaces. *SIGMOD Record*, 29(2):70–81, 2000.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Record*, 27(2):94–105, 1998.

[5] M. Ahmed and L. Khan. SISC: A text classification approach using semi supervised subspace clustering. In *IEEE International Conference on Data Mining Workshops*, pages 1–6, 2009.

[6] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proceedings of the 2nd International Symposium on Information Theory*, pages 267–281, 1973.

[7] A. Albatineh, M. Niewiadomska-Bugaj, and D. Mihalko. On similarity indices and correction for chance agreement. *Journal of Classification*, 23(2):301–313, 2006.

[8] R. Alexandridis, S. Lin, and M. Irwin. Class discovery and classification of tumor samples using mixture modeling of gene expression data, a unified approach. *Bioinformatics*, 20(16):2545–2552, 2004.

[9] G. Ascoli et al. Petilla terminology: Nomenclature of features of GABAergic interneurons of the cerebral cortex. *Nature Reviews Neuroscience*, 9:557–568, 2008.

[10] L. Avrim and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[11] A. Azevedo and M. Santos. KDD, SEMMA and CRISP-DM: A parallel overview. In *IADIS European Conference on Data Mining*, pages 182–185, 2008.

[12] M. Baghshah and S. Shouraki. Metric learning for semi-supervised clustering using pairwise constraints and the geometrical structure of data. *Intelligent Data Analysis*, 13(6):887–899, 2009.

[13] M. Baghshah and S. Shouraki. Kernel-based metric learning for semi-supervised clustering. *Neurocomputing*, 73(7-9):1352–1361, 2010.

[14] F. Baker and L. Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70:31–38, 1975.

[15] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 11–18, 2003.

[16] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning*, pages 19–26, 2002.

[17] S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *SIAM International Conference on Data Mining*, pages 333–344, 2004.

[18] S. Basu, M. Bilenko, and R. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of the International Conference on Machine Learning*, pages 42–49, 2004.

[19] S. Basu, M. Bilenko, and R.J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68. ACM, 2004.

[20] S. Basu, I. Davidson, and K. Wagstaff, editors. *Constrained Clustering: Advances in Algorithms, Theory and Applications*. Chapman and Hall/CRC, 2009.

[21] E. Beale. *Cluster Analysis*. London: Scientific Control Systems, 1969.

[22] R. Bellman. *Adaptive Control Processes - A Guided Tour*. Princeton University Press, 1961.

[23] R. Benavides-Piccione, F. Sichani, I. Yaez, J. DeFelipe, and R. Yuste. Dendritic size of pyramidal neurons differs among mouse cortical regions. *Cerebral Cortex*, 16(7):990–1001, 2005.

[24] K. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–296. ACM, 2002.

[25] P. Berkhin. A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.

[26] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

[27] C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statistics and Data Analysis*, 41(3-4):561–575, 2003.

[28] M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 11–18, 2004.

[29] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[30] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19–26, 2001.

[31] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100. ACM, 1998.

[32] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[33] C. Böhm, K. Kailing, H. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the 4th International Conference on Data Mining*, pages 27–34, 2004.

[34] A. Bouchachia. Learning with partly labeled data. *Neural Computing and Applications*, 16(1):267–293, 2007.

[35] A. Bouchachia and W. Pedrycz. Data clustering with partial supervision. *Data Mining Knowledge Discovery*, 12(1):47–78, 2006.

[36] N. Bouguila and D. Ziou. A hybrid SEM algorithm for high-dimensional unsupervised learning using a finite generalized Dirichlet mixture. *IEEE Transactions on Image Processing*, 15(9):2657–2668, 2006.

[37] N. Bouguila and D. Ziou. High-dimensional unsupervised selection and estimation of a finite generalized Dirichlet mixture model based on minimum message length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1716–1731, 2007.

[38] S. Boutemedjet, D. Ziou, and N. Bouguila. Model-based subspace clustering of non-Gaussian data. *Neurocomputing*, 73(10-12):1730–1739, 2010.

[39] P. Bradley, K. Bennett, and A. Demiriz. Constrained K-means Clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[40] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees.* Chapman and Hall, 1993.

[41] T. Bullock, M. Bennett, D. Johnston, R. Josephson, E. Marder, and Fields R. The neuron doctrine, redux. *Science*, 310(5749):791–793, 2005.

[42] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[43] B. Cauli, J. Porter, K. Tsuzuki, B. Lambolez, J. Rossier, B. Quenet, and E. Audinat. Classification of fusiform neocortical interneurons based on unsupervised clustering. *PNAS*, 91(11):6144–6149, 1987.

[44] M. Ceccarelli and A. Maratea. Improving fuzzy clustering of biological data by metric learning with side information. *International Journal of Approximate Reasoning*, 47(1):45–57, 2008.

[45] Y. Chan, W. Ching, M.K. Ng, and J. Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37(5):943–952, 2004.

[46] A. Chandel, A. Tiwari, and N. Chaudhari. Constructive semi-supervised classification algorithm and its implement in data mining. In *Proceedings of the 3rd International Conference on Pattern Recognition and Machine Intelligence*, pages 62–67. Springer-Verlag, 2009.

[47] C. Chang and H. Chen. Semi-supervised clustering with discriminative random fields. *Pattern Recognition*, 2012.

[48] J. Chang and D. Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Symposium on Applied Computing*, pages 503–507. ACM, 2002.

[49] O. Chapelle, V. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.

[50] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *10th International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

[51] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. CRISP-DM 1.0 Step-by-step Data Mining Guide. Technical report, The CRISP-DM Consortium, 2000.

[52] N.V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

[53] C. Cheng, A.W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 84–93, 1999.

[54] H. Cheng, K.A. Hua, and K. Vu. Constrained locally weighted clustering. In *Proceedings of the 34th Internacional Conference on Very Large Data Bases*, volume 1, pages 90–101, 2008.

[55] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. In S. Basu, I. Davidson, and K. Wagstaff, editors, *Constrained Clustering Advances in Algorithms Theory and Application*, pages 17–33. CRC Press, 2003.

[56] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[57] T. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[58] F. Cozman, I. Cohen, and M. Cirelo. Semi-supervised learning of mixture models. In *20th International Conference on Machine Learning*, pages 99–106, 2003.

[59] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, 1990.

[60] I. Davidson and S. Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from Data*, pages 1–41, 2007.

[61] I. Davidson and S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Proceedings of the 15th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 59–70, 2005.

[62] I. Davidson and S. Ravi. Clustering with constraints: Feasibility issues and the K-means algorithm. In *SIAM International Conference on Data Mining*. SIAM, 2005.

[63] I. Davidson and S. Ravi. Using instance-level constraints in agglomerative hierarchical clustering: Theoretical and empirical results. *Data Mining and Knowledge Discovery*, 18(2):257–282, 2009.

[64] I. Davidson, K. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. In *10th European Conference on Principle and Practice of Knowledge Discovery in Databases*, pages 115–126. Springer-Verlag, 2006.

[65] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227, 1979.

[66] L. de Nó. La corteza cerebral del ratón. *Trabajos Laboratorio Investigaciones Biológicas*, 20:41–78, 1922.

[67] J. DeFelipe. Sesquicentenary of the birthday of santiago ramón y cajal, the father of modern neuroscience. *Trends in Neurosciences*, 25(9):481–484, 2002.

[68] J. DeFelipe et al. Classification and nomenclature of Cortical GABAergic interneurons. *Nature Reviews Neuroscience*, 2012. Submitted.

[69] A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 809–814, 1999.

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

[71] Z. Deng, K. Choi, F. Chung, and S. Wang. Enhanced soft subspace clustering integrating within-cluster and between-cluster information. *Pattern Recognition*, 43(3):767–781, 2010.

[72] M.M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2009.

[73] E. Dimitriadou, S. Dolničar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(3):137–160, 2002.

[74] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos. Locally adaptive metrics for clustering high dimensional data. *Journal of Data Mining and Knowledge Discovery*, 14:63–97, 2007.

[75] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma. Subspace clustering of high dimensional data. In *SIAM International Conference on Data Mining*, pages 517 – 521, 2004.

[76] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, 1973.

[77] D. Dumitriu, R. Cossart, J. Huang, and R. Yuste. Correlation between axonal morphologies and synaptic input kineptics of interneurons from mouse visual cortex. *Cerebral Cortex*, 17(1):81–91, 2007.

[78] J. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, pages 32–57, 1973.

[79] J. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[80] B. Efron. Bootstrap methods: Another look at the jacknife. *Annals of Statistics*, 7:1–26, 1979.

[81] B. Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Associatio*, 78:316–331, 1983.

[82] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd Internacional Conference of Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[83] B. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics, 5th edition, 2011.

[84] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[85] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. American Association for Artificial Intelligence, 1996.

[86] M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[87] D. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1994.

[88] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[89] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[90] C. Fraley. Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20:270–281, 1998.

[91] A. Frank and A. Asuncion. UCI Machine Learning Repository, 2010. `http://archive.ics.uci.edu/ml`.

[92] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–977, 2007.

[93] J. Friedman and J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society*, 66(4):815–849, 2004.

[94] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[95] E. Fromont, A. Prado, and C. Robardet. Constraint-based subspace clustering. In *Proceedings of the 9th SIAM International Conference on Data Mining*, pages 26–37, 2009.

[96] A. Galanopoulos and S. Ahalt. Codeword distribution for frequency sensitive competitive learning with one-dimensional input data. *IEEE Transactions on Neural Networks*, 7(3):752–756, 1996.

[97] L. Galvani. *Aloysii Galvani: De Viribus Electricitatis In Motu Musculari.* Kessinger Publishing, latin edition, 1792.

[98] G. Gan and J. Wu. A convergence theorem for the fuzzy subspace clustering (FSC) algorithm. *Pattern Recognition*, 41:1939–1947, 2008.

[99] G. Gan, J. Wu, and Z. Yang. A fuzzy subspace algorithm for clustering high dimensional data. In X. Li, O. Zaïane, and Z. Li, editors, *Advanced Data Mining and Applications*, volume 4093 of *Lecture Notes in Computer Science*, pages 271–278. Springer Berlin / Heidelberg, 2006.

[100] P. García-López, V. García-Marín, and M. Freire. The discovery of dendritic spines by Cajal in 1888 and its relevance in the present neuroscience. *Progress in Neurobiology*, 83(2):110–130, 2007.

[101] A. Geva. Hierarchical unsupervised fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 7(6):723–733, 1999.

[102] Z. Ghahramani and M. Beal. Variational inference for Bayesian mixtures of factor analyzers. In *Advances in Neural Information Processing Systems 12*, pages 449–455, 2000.

[103] S. Goil, H. Nagesh, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. In *International Conference on Data Engineering*, 1999.

[104] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., 1989.

[105] S Goldman and Y Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning*, pages 327–334, 2000.

[106] C. Golgi. Sulla struttura della sostanza grigia del cervello. *Gazzetta Medica Italiana*, 6:244–246, 1873.

[107] D. Gondek and T. Hofmann. Non-redundant data clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 75–82. IEEE Computer Society, 2004.

[108] A. Gordon. A review of hierarchical classification. *Journal of the Royal Statistical*, 150:119–137, 1987.

[109] J.C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48, 1986.

[110] M. Graham and D. Miller. Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *IEEE Transactions on Signal Processing*, 54(4):1289–1303, 2006.

[111] N. Grira, M. Crucianu, and N. Boujemaa. Unsupervised and semi-supervised clustering: A brief survey. In *A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

[112] L. Guerra, C. Bielza, V. Robles, and P. Larrañaga. Semi-supervised projected model-based clustering. *Data Mining and Knowledge Discovery*, 2012. Submitted.

[113] L. Guerra, L. McGarry, V. Robles, C. Bielza, P. Larrañaga, and R. Yuste. Comparison between supervised and unsupervised classifications of neuronal cell types: A case study. *Developmental Neurobiology*, 71(1):71–82, 2011.

[114] L. Guerra, V. Robles, C. Bielza, and P. Larrañaga. A comparison of clustering quality indices using outliers and noise. *Intelligent Data Analysis*, 16:703–715, 2012.

[115] L. Guerra, V. Robles, C. Bielza, and P. Larrañaga. Partially labeled data clustering in subspaces. *International Journal on Artificial Intelligence Tools*, 2012. Submitted.

[116] I. Gurrutxaga, J. Muguerza, O. Arbelaitz, J. Pérez, and J. Martín. Towards a standard methodology to evaluate internal cluster validity indices. *Pattern Recognition Letters*, 32(3):505–515, 2011.

[117] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[118] M. Hall and L. Smith. Feature subset selection: A correlation based filter approach. In *Proceedings of the International Conference on Neural Information Processing and Intelligent Information Systems*, pages 855–858. Springer, 1997.

[119] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.

[120] D. Hand, P. Smyth, and H. Mannila. *Principles of Data Mining*. MIT Press, 2001.

[121] J. Handl, J. Knowles, and D. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

[122] H. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14(2):174–194, 1958.

[123] M. Helmstaedter, B. Sakmann, and D. Feldmeyer. L2/3 interneuron groups defined by multiparameter analysis of axonal projection, dendritic geometry and electrical excitability. *Cerebral Cortex*, 19(4):951–962, 2008.

[124] M. Helmstaedter, B. Sakmann, and D. Feldmeyer. Neuronal correlates of local, lateral, and translaminar inhibition with reference to cortical columns. *Cerebral Cortex*, 19(4):926–937, 2008.

[125] M. Helmstaedter, B. Sakmann, and D. Feldmeyer. The relation between dendritic geometry, electrical excitability, and axonal projections of L2/3 interneurons in rat barrel cortex. *Cerebral Cortex*, 19(4):938–950, 2008.

[126] J. Hernando. *Interactive Visualization of Detailed Large Neocortical Circuit Simulations*. PhD thesis, Universidad Politécnica de Madrid, 2011.

[127] P. Hoff. Subset clustering of binary sequences, with an application to genomic abnormality data. *Biometrics*, 61(4):1027–1036, 2005.

[128] P. Hoff. Model based subspace clustering. *Bayesian Analysis*, 1(2):321–344, 2006.

[129] J.H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[130] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition*. Wiley, 1999.

[131] D. Hosmer and S. Lemeshow. *Applied Logistic Regression*. Wiley-Interscience Publication, 2nd edition, 2000.

[132] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[133] I. Inza, P. Larrañaga, R. Blanco, and A. Cerrolaza. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, 31(2):91–103, 2004.

[134] DeFelipe. J. Chapter 17. cortical interneurons: from cajal to 2001. In E.C. Azmitia, J. DeFelipe, E.G. Jones, P. Rakic, and C.E. Ribak, editors, *Changing Views of Cajal's Neuron*, volume 136 of *Progress in Brain Research*, pages 215–238. Elsevier, 2002.

[135] A. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[136] A. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall College Division, 1988.

[137] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[138] L. Jing, M. Ng, and Z. Huang. An entropy weighting K-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1026–1041, 2007.

[139] L. Jing, M. Ng, J. Xu, and J. Huang. Subspace clustering of text documents with feature weighting K-means algorithm. In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 802–812, 2005.

[140] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, 1999.

[141] I. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[142] E.G. Jones. History of cortical cytology. In A. Peters and E.G. Jones, editors, *Cerebral Cortex, vol. 1. Cellular Components of the Cerebral Cortex*, Progress in Brain Research, pages 1–32. Plenum Press, 1984.

[143] K. Kailing, H. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of International Conference on Data Mining*, pages 246–257. SIAM, 2004.

[144] E. Kandel, J. Schwartz, and T. Jessell. *Principles of Neural Science*. McGraw-Hill, 4th edition, 2000.

[145] A. Karagiannis, T. Gallopin, D. Csaba, D. Battaglia, H. Geoffroy, J. Rossier, E. Hillman, J. Staiger, and B. Cauli. Classification of NPY-expressing neocortical interneurons. *The Journal of Neuroscience*, 29(11):3642–3659, 2009.

[146] D. Karlis and E. Xekalaki. Choosing initial values for the EM algorithm for finite mixtures. *Computational Statistics and Data Analysis*, 41:577–590, 2003.

[147] G. Karypis, E. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[148] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley series in Probability and Statistics, 2005.

[149] K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 129–134. AAAI Press, 1992.

[150] J. Kittler. Feature set search algorithms. *Pattern Recognition and Signal Processing*, pages 41–60, 1978.

[151] D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning*, pages 307–314, 2002.

[152] D. Kleinbaum, L. Kupper, and L. Chambless. Logistic regression analysis of epidemiologic data: Theory and practice. *Communications on Statistics*, 11(5):485–547, 1982.

[153] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[154] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[155] T. Kohonen. *Self-Organizing Maps*. Spriger-Verlag, 2001.

[156] I. Kononenko. Semi-naive Bayesian classifier. In *Proceedings of the European Working Session on Learning on Machine Learning*, pages 206–219. Springer-Verlag New York, 1991.

[157] S. Kotsiantis, I. Zaharakis, and P. Pintelas. Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190, 2006.

[158] J. Kozloski, F. Hamzei-Sichani, and R. Yuste. Stereotyped position of local synaptic targets in neocortex. *Science*, 293:868–872, 2001.

[159] H. Kriegel, P. Kroger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 250–257. IEEE Computer Society, 2005.

[160] H. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–58, 2009.

[161] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[162] T. Lange, M. Law, A. Jain, and J. Buhmann. Learning with constrained and unlabelled data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 731–738, 2005.

[163] M. Law, M. Figueiredo, and A. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

[164] W. Lee, S. Stolfo, and K. Mok. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, 14:533–567, 2000.

[165] J. Li and H. Zha. Two-way Poisson mixture models for simultaneous document classification and word clustering. *Computational Statistics*, 50(1):163–180, 2006.

[166] Y. Li, M. Dong, and J. Hua. A Gaussian mixture model to detect clusters embedded in feature subspace. *Journal of Communications in Information and Systems*, 7(4):337–352, 2007.

[167] Y. Li, M. Dong, and J. Hua. Simultaneous localized feature selection and model detection for Gaussian mixtures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):953–960, 2009.

[168] T. Lin, J. Lee, and S. Yen. Finite mixture modelling using the Skew normal distribution. *Statistica Sinica*, 17:909–927, 2007.

[169] B. Liu, Y. Xia, and P. Yu. Clustering through decision tree construction. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 20–29. ACM, 2000.

[170] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 1250–1254, 2007.

[171] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[172] H. Liu, H. Motoda, R. Setiono, and Z. Zhao. Feature selection: An ever evolving frontier in data mining. In *Proceedings of the 4th International Workshop on Feature Selection in Data Mining*, pages 4–13, 2010.

[173] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17:491–502, 2005.

[174] Y. Liu, R. Jin, and A. Jain. BoostCluster: Boosting clustering by pairwise constraints. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 450–459, 2007.

[175] S.P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[176] L. Lovmar, A. Ahlford, M. Jonsson, and A.C. Syvanen. Silhouette scores for assessment of SNP genotype clusters. *BMC Genomics*, 6(1):1–35, 2005.

[177] Z. Lu and T.K. Leen. Semi-supervised learning with penalized probabilistic clustering. *Advances in Neural Information Processing Systems*, 17:849–856, 2005.

[178] L. Luciani. *Fisiologia dell'Uomo volume 2*. Societa Editrice Libraria, 1905.

[179] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[180] S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[181] R. Maitra. Initializing partition-optimization algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6:144–157, 2009.

[182] R. Maitra and V. Melnykov. Simulating data to study performance of finite mixture modeling and clustering algorithms. *Journal of Computational and Graphical Statistics*, 19:354–376, 2010.

[183] P. Mallapragada, J. Rong, A. Jain, and Y. Liu. SemiBoost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009.

[184] E. Marin, G. Jefferys, T. Komiyama, and H. Zhu. Representation of the glomerular olfactory map in the drosophila brain. *Cell*, 149:243–255, 2002.

[185] H. Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006.

[186] M.M. Masud, Jing Gao, L. Khan, Jiawei Han, and B. Thuraisingham. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 929–934, 2008.

[187] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[188] W. McCulloch and W. Pitts. A logical calculus of ideas imminet in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[189] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics, 2000.

[190] G. McLachlan, G. Peel, K. Basford, and P. Adams. Fitting of mixtures of normal and t-components. *Journal of Statistical Software*, 4(2):909–927, 1999.

[191] V. Melnykov and I. Melnykov. Initializing the EM algorithm in Gaussian mixture models with an unknown number of components. *Computational Statistics & Data Analysis*, 56(6):1381–1395, 2012.

[192] D. Miller and J. Browning. A mixture model and EM-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1468–1483, 2003.

[193] G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[194] M. Minsky. Steps toward artificial intelligence. In *Computers and Thought*, pages 406–450. McGraw-Hill, 1961.

[195] T. Mitchell. *Machine Learning*. McGraw-Hill, 1st edition, 1997.

[196] G. Moise, J Sander, and M. Ester. Robust projected clustering. *Knowledge and Information Systems*, 14(3):273–298, 2008.

[197] G. Moise, A. Zimek, P. Kröger, H. Kriegel, and J. Sander. Subspace and projected clustering: Experimental evaluation and analysis. *Knowledge and Information Systems*, 21(3):299–326, 2009.

[198] V. Mountcastle. Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of Neurophysiology*, 20:408–442, 1957.

[199] V. Mountcastle. The columnar organization of the cerebral cortex. *Brain*, 120:701–722, 1997.

[200] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. In *Proceedings of 35th International Conference on Very Large Data Bases*, volume 2, pages 1270–1281. VLDB Endowment, 2009.

[201] K. Müller, S. Mika, G. Rä, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[202] T. Ng, T. Pham, and X. Jia. Feature interaction in subspace clustering using the Choquet integral. *Pattern Recognition*, 45(7):2645–2660, 2011.

[203] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134, 2000.

[204] N. Pal and J. Bezdek. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):549–557, 1993.

[205] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter - Special Issue on Learning From Imbalanced Datasets*, 6(1):90–105, 2004.

[206] D. Pascual, F. Pla, and J. Sánchez. Cluster validation using information stability measures. *Pattern Recognition Letters*, 31(6):454–461, 2010.

[207] A. Patrikainen and H. Mannila. Subspace clustering of high-dimensional binary data - a probabilistic approach. In *SIAM International Conference on Data Mining. Workshop on Clustering High Dimensional Data and its Applications*, pages 57–65, 2004.

[208] A. Patrikainen and M. Meila. Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):902–916, 2006.

[209] M. Pazzani. Searching for dependencies in Bayesian classifiers. In *Artificial Intelligence and Statistics, Lecture Notes in Statistics*, pages 239–248. Springer-Verlag, 1996.

[210] J. Peña, J. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the K-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[211] K. Pearson. Notes on the history of correlation. *Biometrika*, 13(1):25–45, 1920.

[212] A. Peters. Number of neurons and synapses in primary visual cortex. *Cerebral Cortex*, 6(1):267–294, 1987.

[213] N. Pise and P. Kulkarni. A survey of semi-supervised learning methods. In *International Conference on Computational Intelligence and Security*, volume 2, pages 30–34, 2008.

[214] C. Procopiuc, M. Jones, P. Aggarwal, and T. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM International Conference on Management of Data*, 2002.

[215] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[216] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011.

[217] S. Ramón y Cajal. Estructura de los centros nerviosos de las aves. *Revista Trimestral de Histología Normal y Patológica*, 1:1–10, 1888.

[218] S. Ramón y Cajal. *Manual de Histología Normal y Técnica Micrográfica*. Aguilar, 1889.

[219] S. Ramón y Cajal. La rétine des vertébrés. *La Cellule*, 9(1):119–257, 1893.

[220] S. Ramón y Cajal. El azul de metileno en los centros nerviosos. *Revista Trimestral Micrográfica*, 1:151–203, 1896.

[221] S. Ramón y Cajal. Las espinas colaterales de las células del cerebro teñidas por el azul de metileno. *Revista Trimestral Micrográfica*, 1:123–126, 1896.

[222] S. Ramón y Cajal. *Textura del Sistema Nervioso del Hombre y de los Vertebrados*. Moya, 1899.

[223] S. Ramón y Cajal. Recreaciones estereocópicas y binoculares. *La Fotografia*, 27:41–48, 1901.

[224] S. Ramón y Cajal. *Recuerdos de mi Vida*. Pueyo, 3rd edition, 1923.

[225] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[226] E. Rendón, I. Abundez, A. Arizmendi, and E. Quiroz. Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, 5(1):27–34, 2011.

[227] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *7th IEEE Workshop on Applications of Computer Vision*, pages 29–36, 2005.

[228] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.

[229] P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[230] C. Ruiz, M. Spiliopoulou, and E. Menasalvas. Density-based semi-supervised clustering. *Data Minining and Knowledge Discovery*, 21(3):345–370, 2010.

[231] D. Rumerlhart, G. Hinton, and R. Williams. Learning internal representations by backpropagation errors. *Nature*, 323:533–536, 1986.

[232] P. Russel and T. Rao. On habitat and association of species of Anopheline Larvae in south-eastern Madras. *Journal of Malaria Institute India*, 3:153–178, 1940.

[233] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[234] G. Schwarz. Estimating the dimensions of a model. *Annals of Statistics*, 6:381–396, 1978.

[235] K. Sequeira and M. Zaki. SCHISM: a new approach to interesting subspace mining. *International Journal of Business Intelligence and Data Mining*, 1(2):137–160, 2005.

[236] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems 16*, pages 1–8. MIT Press, 2003.

[237] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*, pages 1–66, 2012.

[238] P. Sneath. The application of computers to taxonomy. *Journal of General Microbiology*, 17:201–226, 1957.

[239] R. Sokal and C. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[240] T. Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter*, 5:1–34, 1948.

[241] H. Steinhaus. Sur la division des corp materiels en parties. *Bulletin of the Polish Academy of Sciences*, 4(12):801–804, 1956.

[242] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B*, 36:111–147, 1974.

[243] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[244] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.

[245] F. Tan, X. Fu, Y. Zhang, and A. Bourgeois. A genetic algorithm-based method for feature subset selection. *Soft Computing*, 12(2):111–120, 2007.

[246] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition, 2009.

[247] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[248] A. Tsiola, F. Hamzei-Sichani, Z. Peterlin, and R. Yuste. Quantitative morphological classification of layer 5 neurons from mouse primary visual cortex. *The Journal of Comparative Neurology*, 461:415–428, 2003.

[249] C. Van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[250] V. Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, 1995.

[251] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[252] L. Vendramin, R. Campello, and E. Hruschka. On the comparison of relative clustering validity criteria. In *Proceedings of the 9th SIAM International Conference on Data Mining*, pages 733–744, 2009.

[253] N. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[254] K. Wagstaff, S. Basu, and I. Davidson. When is constrained clustering beneficial, and why? In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 58, pages 59–62, 2006.

[255] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1103–1110, 2000.

[256] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained K-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers, 2001.

[257] H. Waldeyer. Ueber einige neuere forschungen im gebiete der anatomie des centralnervensystems. *Deutsche medicinische Wochenschrift*, 17:1213–1218, 1244–1246, 1287–1289, 1331–1332, 1350–1356, 1891.

[258] F. Wang, C. Zhang, H.C. Shen, and J. Wang. Semi-supervised classification using linear neighborhood propagation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 160–167, 2006.

[259] J.H Ward. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

[260] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[261] R. Wirth. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Application of Knowledge Discovery and Data Mining*, pages 29–39, 2000.

[262] I. Witten, E. Frank, and M. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[263] H. Wold. *Encyclopedia of Statistical Sciences*, volume 6, chapter Partial least squares, pages 581–591. Wiley, 1985.

[264] A. Wong, J. Wang, and R. Axel. Spatial representation of the glomerular map in the Drosophila protocerebrum. *Cell*, 109:229–241, 2002.

[265] K. Woo, J. Lee, M. Kim, and Y. Lee. FINDIT: A fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[266] J. Wu, H. Yuan, H. Xiong, and G. Chen. Validation of overlapping clustering: A random clustering perspective. *Information Sciences*, 180(22):4353–4369, 2010.

[267] Y. Xia. A global optimization method for semi-supervised clustering. *Data Mining and Knowledge Discovery*, 18(2):214–256, 2009.

[268] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512, 2002.

[269] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[270] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2):44–49, 1998.

[271] M. Yang, C. Lai, and C. Lin. A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognition*, 2012.

[272] K. Yeung, D. Hayunor, and W. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

[273] X. Yin, S. Chen, E. Hu, and D. Zhang. Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333, 2010.

[274] K. Yip, D. Cheung, and M. Ng. HARP: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16:1387–1397, 2004.

[275] K. Yip, D. Cheung, and M. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. *International Conference on Data Engineering*, pages 329–340, 2005.

[276] K. Yip, L. Cheung, D. Cheung, L. Jing, and M. Ng. A semi-supervised approach to projected clustering with applications to microarray data. *International Journal of Data Minining and Bioinformatics*, 3(3):229–259, 2009.

[277] M.L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):176–189, 2005.

[278] L. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[279] X. Zhang, Y. Qiu, and Y. Wu. Exploiting constraint inconsistence for dimension selection in subspace clustering: A semi-supervised approach. *Neurocomputing*, 74(17):3598–3608, 2011.

[280] X. Zhang, Y. Wu, and Y. Qiu. Constraint based dimension correlation and distance divergence for clustering high-dimensional data. In *IEEE 10th International Conference on Data Mining*, pages 629–638, 2010.

[281] S. Zhong and J. Ghosh. Scalable, model-based balanced clustering. In *SIAM International Conference on Data Mining*, pages 71–82, 2003.

[282] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1003, 2003.

[283] X. Zhu. Semi-Supervised Learning Literature Survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

[284] X. Zhu and Z. Ghahramani. Learning from Labeled and Unlabeled Data with Label Propagation. Technical report, School of Computer Science, Carnegie Mellon University, 2002.

[285] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.