UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO DI MATEMATICA E INFORMATICA
DOTTORATO DI RICERCA IN MATEMATICA E INFORMATICA

- XXII CICLO -

Algorithms for Internal Validation Clustering Measures in the Post Genomic Era

Author

Filippo Utro

Coordinator

Prof. Camillo Trapani

Thesis Advisor
Prof. Raffaele Giancarlo

Settore Scientifico Disciplinare INF/01

Reviewers

Prof. Concettina Guerra
College of Computing, Georgia Tech, GA, USA
Dipartimento di Ingegneria Informatica, Università di Padova, Italy

Prof. Paola Sebastiani
Department of Biostatistics, Boston University School of Public Heath, MA, USA

Algorithms for Internal Validation Clustering Measures in the Post Genomic Era

Abstract: Inferring cluster structure in microarray datasets is a fundamental task for the so-called -omic sciences. It is also a fundamental question in Statistics, Data Analysis and Classification, in particular with regard to the prediction of the number of clusters in a dataset, usually established via internal validation measures. Despite the wealth of internal measures available in the literature, new ones have been recently proposed, some of them specifically for microarray data.

In this dissertation, a study of internal validation measures is given, paying particular attention to the stability based ones. Indeed, this class of measures is particularly prominent and promising in order to have a reliable estimate of the correct number of clusters in a dataset. For this kind of measures, a new general algorithmic paradigm is proposed here that highlights the richness of measures in this class and accounts for the ones already available in the literature. Moreover, some of the most representative data-driven validation measures are also considered. Extensive experiments on twelve benchmark microarray datasets are performed, using both Hierarchical and K-means clustering algorithms, in order to assess both the intrinsic ability of a measure to predict the correct number of clusters in a dataset and its merit relative to the other measures. Particular attention is given both to precision and speed. The main result is a hierarchy of internal validation measures in terms of precision and speed, highlighting some of their merits and limitations not reported before in the literature. This hierarchy shows that the faster the measure, the less accurate it is. In order to reduce the time performance gap between the fastest and the most precise measures, the technique of designing fast approximation algorithms is systematically applied. The end result is a speed-up of many of the measures studied here that brings the gap between the fastest and the most precise within one order of magnitude in time, with no degradation in their prediction power. Prior to this work, the time gap was at least two orders of magnitude.

Finally, for the first time in the literature, a benchmarking of Non-negative Matrix Factorization as a clustering algorithm on microarrays is provided. Such a benchmarking is novel and sheds further light on the use of Non-negative Matrix Factorization as a data mining tool in bioinformatics. Given the increasing popularity of Non-negative Matrix Factorization for data mining in biological data, the results reported here seem to contribute to the proper use of the technique, being well aware of its limitations, in particular the extensive use of computational resources it needs.

Keywords: Algorithms and Data Structures, Experimantal Analysis of Algorithms, General Statistics, Analysis of Massive Datasets, Machine Learning, Computational Biology, Bioinformatics.

Acknowledgments

I owe a great deal of thanks to many people for making this thesis possible. First, I would like to express my gratitude for my advisor Prof. Raffaele Giancarlo, who has been leading and supporting me and my research to be fruitful in his patience.

A huge thanks goes to Davide Scaturro for his collaboration in the first stage of my work. Without his skillful support my projects would not have been possible.

I truly appreciate to Dr. Giusi Castiglione that read drafts of the thesis and offered important suggestions for improvement and more important her friendship.

Thanks to my fellow PhD friends, in particular Fabio Bellavia, Marco Cipolla, Filippo Millonzi and Luca Pinello for our broad-ranging discussions and for sharing the joys and worries of academic research.

Furthermore I am deeply indebted to my colleagues at Department of Mathematics and Computer Science that have provided the environment for sharing their experiences about the problem issues involved as well as participated in stimulating team exercises developing solutions to the identified problems. I would specially like to thank Dr. Giosué Lo Bosco and Prof. Marinella Sciortino for their extremely valuable experiences, support, insights and more important their friendship.

Finally, I wish to express my gratitude to my family and friends who provided continuous understanding, patience, love and energy. In particular, I would like to express a heartfelt thanks to my parents and my girlfriend Marcella for their infinite support in my research endeavors.

Thanks to all of you.

Originality Declaration

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the test. I give consent to this copy of my thesis, when deposited in the University Library, begin available for loan and photocopying.

Signed .... January 2011

Contents

List of Figures

List of Tables

Introduction

In the past 15 years, a paradigm shift in Life Sciences research has taken place, thanks to the availability of genomic and proteomic data on an unprecedented scale. Such a revolutionary change has posed new challenges to Mathematics, Statistics and Computer Science, since the conceptual tools proper of those three disciplines are fundamental for the study of biological questions via computational tools, on a genomic scale. In the following, by way of example, the converging views of Hood and Galas, two authorities in the Life Sciences, and Knuth, an authority in Computer Science, are summarized.

In the Fifty Years Commemorative Issue of Nature on the discovery of DNA, Hood and Galas [81] conclude their contribution by clearly stating that a mathematical notion of "biological information" readily usable for the development of computational tools for -omic investigations is lacking and that such a notion would be a fundamental contribution of the Exact Sciences to the Life Sciences. Moreover, Knuth [11] assert that the contributions given by Mathematics, Statistics and Computer Science to Molecular Biology would have been unpredictable, both in depth and breadth, only at the end of the 90s. Yet, those contributions are only a small fraction of the challenges faced by the Computer and Information Sciences in strategic junctions of this domain. Therefore, the development of a new core area of Mathematics and Computer Science is taking place. Although the topics discussed in this dissertation are classical, they will be developed having well in mind such an important new direction. In particular, this dissertation focuses on various aspects of clustering when used in conjunction with microarray data.

Microarrays are a useful and, by now, well established technology in genomic investigation. Indeed, experiments based on that technology are increasingly being carried out in biological and medical research to address a wide range of problems, including the classification of tumors [8, 10, 48, 65, 135, 138, 146], where a reliable and precise classification is essential for successful diagnosis and treatment. By allowing the monitoring of gene expression levels on a genomic scale, microarray experiments may lead to a more complete understanding of the molecular variations among tumors and hence to a finer and more reliable classification. An important statistical problem associated with tumor classification is the identification of new tumor classes using gene expression profiles, which has revived interest in cluster analysis. However, the novelty, noisiness and high dimensionality of microarray data provide new challenges even to a classic and well studied area such as clustering. More in general, new methodological and computational challenges are proposed daily [47, 48, 96, 115]. As a results, there has been a "malthusian growth" of new statistical and computational methods for genomic analysis. Unfortunately, many papers for -omic research describe development or application of statistical methods for microarray data that are questionable [118]. In view of this latter peril, in this dissertation an effort has been made to use a methodology that is sound and coherent for the experimental validation of the computational methods proposed here.

In microarray data analysis there are two essential aspects of clustering: finding a "good" partition of the datasets and estimating the number of clusters, if any, in a dataset. The former problem is usually solved by the use of a clustering algorithm. In the Literature, a large number of clustering algorithms has been proposed and many of these have been applied to genomic data [42], the most famous are: K-means [87], fuzzy c-means, self-organizing maps [136, 148, 153], hierarchical clustering [87], and model-based clustering [154, 155]. Some of those studies concentrate both on the ability of an algorithm to obtain a high quality partition of the data and on its performance in terms of computational resources, mainly CPU time (see [27, 55, 100, 150] and references therein).

However, the most fundamental issue is the latter problem, i.e., the determination of the number of clusters. Despite the vast amount of knowledge available in the general data mining literature, e.g., [29, 49, 67, 68, 79, 87, 93, 95], gene expression data provide unique challenges, in particular with respect to internal validation indices. Indeed, they must predict how many clusters are really present in a dataset, an already difficult task, made even worse by the fact that the estimation must be sensible enough to capture the inherent biological structure of functionally related genes. Despite their potentially important role, both the use of classic internal validation indices and the design of new ones, specific for microarray data, do not seem to have great prominence in bioinformatics, where attention is mostly given to clustering algorithms. The excellent survey by Handl et al. [73] is a big step forward in making the study of those techniques a central part of both research and practice in bioinformatics, since it provides both a technical presentation as well as valuable general guidelines about their use for post-genomic data analysis. Although much remains to be done, it is, nevertheless, an initial step.

For instance, in the general data mining literature, there are several studies, e.g., [119], aimed at establishing the intrinsic, as well as the relative, merit of an index. To this end, the two relevant questions are:

(i) What is the precision of an index, i.e., its ability to predict the correct number of clusters in a dataset? That is usually established by comparing the number of clusters predicted by the index against the number of clusters in the true solution of several datasets, the true solution being a partition of the dataset in classes that can be trusted to be correct, i.e., distinct groups of functionally related genes.

(ii) Among a collection of indices, which is more accurate, less algorithm dependent, etc.,?.

Precision versus the use of computational resources, primarily execution time, would be an important discriminating factor.

Contributions and Thesis Outline

From the previous brief description of the state of the art it is evident that, for the special case of microarray data, the experimental assessment of the "fitness" of a measure has been rather ad hoc and studies in that area provide only partial and implicit comparison among measures. Moreover, contrary to research in the clustering literature, the performance of validation methods in terms of computational resources, again mainly CPU time, is hardly assessed both in absolute and relative terms. This dissertation is an attempt to tackle the stated limitations of the state of the art in a homogeneous way. It is organized as follows:

- Chapter 1 provides the background information relevant to this thesis. Indeed, a formal definition of the clustering problem and the notation used in this dissertation is given. Moreover, the two main classes of clustering algorithms proposed in the Literature, some methods for the assessment and evaluation of cluster quality, and data generation/perturbation methods, are also detailed.

- Chapter 2 provides a presentation of some basic validation techniques. In detail, external and internal indices are outlined. In particular, three external indices that assess the agreement between two partitions are presented. Moreover, four internal measures useful to estimate the correct number of clusters present, if any, in a dataset are detailed. They are based on: compactness, hypothesis testing in statistics and jackknife techniques. The described measures have been selected since they are particularly prominent in the clustering literature. Moreover, they are all characterized by the fact that, for their prediction, they make use of nothing more than the dataset available, i.e., they are all data-driven. It is worth pointing out that another class of measure is based on Bayesian Model and it is an aproach both to cluster analysis and the estimation of the "best" number of clusters in a dataset. In order to keep this thesis focus it will not be discussed in this dissertation. The interested reader is referred to [141] and references therein for an in depth treatment of the relevant topics regarding Bayesian Model Based Clustering.

- Chapter 3 provides one of the main topics of this thesis. Indeed, for the first time in the Literature, a general algorithmic paradigm of stability internal validation measure is introduced. It can be seen as a generalization of earlier works by Breckenridge and Valentini. Moreover, it is shown that each of the known stability based measures is an instance of such a novel paradigm. Surprisingly, also the Gap Statistics falls within the new paradigm. Moreover, from this general algorithmic paradigm it is simple to design new stability internal measure combining the building blocks of the measures detailed. As will be evident in this dissertation, this particular category of internal validation measure obtains excellent results in terms of estimation of number of clusters in a dataset. In fact, prior to this study, they were perceived as a most promising avenue of research in the development of internal validation measures. Therefore, the identification of an algorithmic paradigm describing the entire class seems to be a substantial methodological contribution to that area.

- Chapter 4 provides a formal description of one of the methodologies that has gained prominence in the data analysis literature: Non-negative Matrix Factorization. In particular, of relevance for this thesis, is the use of Non-negative Matrix Factorization as a clustering algorithm.

- Chapter 5 describes the experimental methodology used in this thesis. The experimental setup include, to the best of our knowledge, the most complete representative collection of datasets used in the Literature. In particular, this collection is composed of nine microarray datasets that seem to be a *de facto* standard in the specialistic literature and three artificial dataset generated in order to evaluate specific aspects of the clustering methodology, when used on microarray data. Moreover, this chapter provides an exhaustive study of the three external indices detailed in this dissertation. Furthermore, a benchmarking of Non-negative Matrix Factorization as a clustering algorithm on microarrays data. Such a benchmarking is novel and sheds further light on the use of Non-negative Matrix Factorization as a data mining tool in bioinformatics.

- Chapter 6 provides a benchmarking of some of the most prominent internal validation measures in order to establish the intrinsic, as well as the relative, merit of a measure taking into account both its predictive power and its computational demand. This benchmarking shows that there is a natural hierarchy, in terms of the trade-off time/precision, for the measures taken into account. That is, the faster the measure, the less accurate it is. Although this study has been published only recently, it is already being referenced, even “back to back” to fundamental studies on clustering such as the papers by D’haeseleer [42] and Handl et al. [73].

- Chapter 7, based on the benchmarking in Chapter 6, investigates systematically the application of the idea of algorithmic approximation to internal validation measures in order to obtain speedups. That is, the development of new methods that closely track the behavior of existing methods, but that are substantially faster in time. Such a systematic study seems to be quite novel. In this chapter, several approximation algorithms and two general approximation schemes are proposed. In particular, an approximation of a “star” of the area as Gap Statistics is proposed and it is shown that it grants clearly superior results. Indeed, depending on the dataset, it is from two to three orders of magnitude faster than the Gap Statistics, with a better prediction of the correct number of clusters in a datasets. Finally, an approximation of Consensus Clustering it is also proposed. In terms of the trade-off time/precision, it turns out to be the best among all measures studied in this dissertation. Even more remarkably, it reduces the time performance gap between the fastest measures and the most precise to one order of magnitude. Prior to this work, the gap was at least two orders of magnitude.

- Chapter 8 offers some conclusions as well as some future lines of research for further development of the ideas presented in this dissertation.

CHAPTER 1

Background on Cluster Analysis

In this chapter, some fundamental aspects of cluster analysis are presented. In particular, the two main classes of clustering algorithms proposed in the literature are described. Moreover, methods for the assessment and evaluation of cluster quality are discussed as well as data generation/perturbation methods which can be applied to the former.

1.1 Basic Mathematical Problem Formulations

Consider a set of n items $\Sigma = \{\sigma_1, \dots, \sigma_n\}$, where σ_i, with $1 \leq i \leq n$, is defined by m numeric values, referred to as features or conditions. That is, each σ_i is an element in a m-dimensional space. Let $\mathcal{C}_k = \{c_1, c_2, \dots, c_k\}$ be a *partition* of Σ, i.e., a set of subsets of Σ such that $\bigcup_{i=1}^{k} c_i = \Sigma$ and $c_i \cap c_j = \emptyset$ for $1 \leq i \neq j \leq k$. Each subset c_i, where $1 \leq i \leq k$, is referred to as a *cluster*, and $\mathcal{C}_k$ is referred to as a *clustering solution*. The aim of cluster analysis is to determine a partition of Σ according to a similarity/distance S, which is referred to as *similarity/distance metric*. It is defined on the elements in Σ. In particular, one wants that items in the same cluster have "maximal similarity", while items in different clusters are "dissimilar". For instance, an example comes from molecular data analysis [158], in which a set of genes are the items and the features are the expression level measurements in m different experimental conditions or in m different time periods. Clustering would highlight groups of genes that are, for instance, functionally correlated or that have the same response to medical treatments.

Usually, the set Σ containing the items to be clustered is represented in one of two different ways: (1) a data matrix D, of size $n \times m$, in which the rows represent the items and the columns represent the condition values; (2) a similarity/dissimilarity matrix S, of size $n \times n$, in which each entry $S_{i,j}$, with $1 \leq i \neq j \leq n$, is the value of similarity/dissimilarity of the pair (i, j) of items. Specifically, the value of $S_{i,j}$ can be computed using rows i and j of D. Hence, S can be derived from D, but not viceversa. The specification and formalization of a similarity metric, via mathematical functions, depends heavily on the application domain and it is one of the key steps in clustering, in particular in the case of microarray data. The state of the art, as well as some relevant progress in the identification of good distance functions for microarrays, is presented in [139].

1.2 Clustering Algorithms

Usually, the partition of the items in Σ is accomplished by means of a clustering algorithm A. In this chapter, only the class of clustering algorithms that takes as input D and an integer k and return a partition $\mathcal{C}_k$ of Σ into k subsets is taken in account. There is a rich literature about clustering algorithms, and there are many different classifications of them [87, 93]. A survey of classic as well as more innovative clustering algorithms, specifically designed for microarray data, is given in [152]. A classical classification is *hierarchical* versus *partitional* algorithms. The hierarchical clustering algorithms produce a partition by a nested sequence of partitions and they are outlined in Section 1.2.1, where three of them are detailed [87], referred to as Average Link (Hier-A for short), Complete Link (Hier-C for short), and Single Link (Hier-S for short). The partitional clustering algorithms directly decompose the dataset into a partition $\mathcal{C}_k$. One of the most prominent in that class, i.e., K-means [87], is detailed in Section 1.2.2.

1.2.1 Hierarchical Algorithms

In hierarchical clustering, items are related by a tree[1] structure, referred to as *dendogram* such that similar items are at the leaves of the same subtree. Each internal node represents a cluster and the leaves correspond to the items. These algorithms can be either agglomerative ("bottom-up"), in which one starts at the leaves and successively merges clusters together; or divisive ("top-down") in which one starts at the root and recursively splits the clusters. For instance, in Fig. 1.1, a dendogram obtained by a run of Hier-A on a dataset of 24 elements according to the Euclidean distance is given. The partition $\mathcal{C}_4$ is marked. In what follows, only the agglomerative approach is outlined. In particular, the Hier-A, Hier-C and Hier-S clustering algorithms are considered in this thesis. Each of them is an instance of the HIERARCHICAL paradigm described in Fig. 1.2 as a procedure. The interested reader is referred to [52, 77, 87] for an in depth treatment of the hierarchical clustering algorithms.

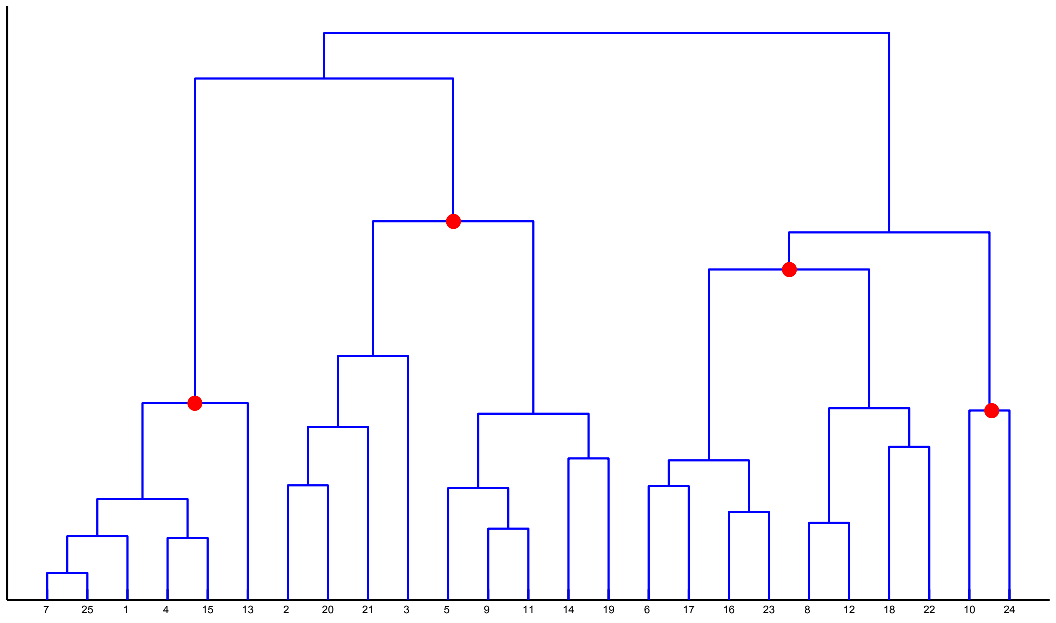

Figure 1.1: Example of a dendogram. The nodes in red indicate the partition $\mathcal{C}_4$ where the leafs (e.g. items) in the same subtree are in the same cluster.

HIERARCHICAL takes as input a data matrix D and the desired number k of clusters. In step 1, a similarity matrix S is computed from the data matrix D. In step 2, each item is considered as a cluster, i.e., this step corresponds to the leaf level of the dendogram.

[1]One assumes that the reader is familiar with that general concepts of graph theory such as trees and planar graphs. The reader is referred to standard references for an appropriate background [26].

```
HIERARCHICAL($D, k$)

 1. Compute a similarity matrix $S$
 2. Initialize each item as a cluster
 for $i \leftarrow 1$ to $n - k$ do
   begin
 3.   Select the two most "similar" clusters
 4.   Merge them.
   end
 return ($clustering\_solution$)
```

Figure 1.2: The HIERARCHICAL procedure.

One single iteration of the **for** loop is discussed, which is repeated until $n - k$ steps are performed, i.e., until k clusters are obtained. In step 3, the most "similar" clusters c_i and c_j are selected. The "similarity" between two clusters is measured by a distance function *Dist*. Therefore, if c_i and c_j are the most similar $Dist(c_i, c_j)$ is minimum. In step 4, c_i and c_j are merged. Finally, the clustering solution obtained is given as output.

The three hierarchical clustering algorithms differ one from the other only for the distance function used to select the two clusters in step 3.
Hier-A computes the distance between two clusters c_i and c_j as the average of the values of the similarity metric between the items of c_i and c_j, respectively. Formally:

$$Dist(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{x \in c_i} \sum_{y \in c_j} S_{x,\, y}$$

where $|c_i|$ and $|c_j|$ are the sizes of the clusters c_i and c_j, respectively.
Hier-C computes the distance between two clusters as the maximal item-to-item similarity matric value. Formally:

$$Dist(c_i, c_j) = \max_{x \in c_i,\, y \in c_j} S_{x,\, y}.$$

Finally, in Hier-S, the distance between two clusters is computed as the minimal item-to-item similarity matric value Formally:

$$Dist(c_i, c_j) = \min_{x \in c_i,\, y \in c_j} S_{x,\, y}.$$

1.2.2 Partitional Algorithms

The goal of partitional clustering algorithms is to decompose directly the dataset into a set of disjoint clusters, obtaining a partition which should optimize a given objective function. Intuitively, the criteria one follows are to minimize the dissimilarity between items in the same cluster and to maximize the dissimilarity between items of different clusters. Therefore, clustering can be seen as an optimization problem, where one tries to minimize/maximize an objective function. Unfortunately, it can be shown to be NP-Hard [57]. For completeness, the number of possible partitions can be computed via the Stirling numbers of the second kind [87]:

$$\frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n.$$

Even for small k and n, it is such a substantially large number to discourage exhaustive search. Therefore, existing algorithms provide different heuristics to solve the various versions of clustering as an optimization problem [75]. Here, K-means [113] is detailed. The interested reader is referred to [87, 93] for an in depth treatment of the partitional clustering algorithm.

In K-means, a cluster is represented by its "center", which is referred to as *centroid*. The aim of the algorithm is to minimize a squared error function, i.e., an indicator of the distance of the n data points from their respective cluster centers. Formally:

$$\sum_{j=1}^{k}\sum_{x\in c_j}\|x-\overline{c_j}\|^2,$$

where $\overline{c_j}$ is the centroid of cluster c_j.

The procedure summarizing K-means is reported in Fig. 1.3. In addition to the input parameters of the HIERARCHICAL procedure, K-MEANS takes also the maximum number of iterations allowed to the algorithm, which is referred to as $Niter$.

In steps 1 and 2, the dataset D is partitioned into k clusters, by a random selection of the k centroids. One single iteration of the **while** loop is discussed, which is repeated until at least one of the following two conditions is satisfied: (i) the clustering solution has not changed or (ii) the maximum number of iterations has been reached. The former condition is indicates in the procedure with H, while the latter condition prevents endless oscillations [87]. The main part of the algorithm consists of steps 5-7, where each item in D is assigned to the centroids with minimum distance. Indeed, for each $i \in D$, with $1 \leq i \leq n$, the distance between i and each centroid is computed in step 5. In step 6, item i is assigned to the cluster whose distance from i is minimal. Finally, the clustering solution is given as output. Figs. 1.4(b)-(d) report an example of successive iterations of K-means, where the points in red are the two centroids. The algorithm is applied to the dataset, with two well-separated clusters, reported in Fig. 1.4(a).
Moreover, it is worth pointing out that a clustering solution obtained by another clustering algorithm (e.g. hierarchical) can be used as an initial clustering solution, instead of the random partition generated in steps 1 and 2, by the K-MEANS procedure. Accordingly, when K-MEANS starts the clustering from a random partition it is referred to as K-means-R, while when it starts from an initial partition produced by one of the chosen hierarchical methods it is referred to as K-means-A, K-means-C and K-means-S, respectively.

1.3 Assessment of Cluster Quality: Main Problems Statement

In bioinformatics, a sensible biological question would be, for instance, to find out how many functional groups of genes are present in a dataset. Since the presence of "statistically significant patterns" in the data is usually an indication of their biological relevance [109], it makes sense to ask whether a division of the items into groups is statistically significant.

```
K-MEANS($D, k, Niter$)
1. Extract, at random, $k$ items from $D$ and use them as centroids
2. Assign each item of $D$ to the centroid with minimum distance
3. $step \leftarrow 0$
while ($H$ or $step > Niter$) do
  begin
4.  $step \leftarrow step + 1$
    for each $i \in D$ do
      begin
5.      Compute the distance between $i$ and each centroid
6.      Assign $i$ to the centroid with minimum distance
7.      Compute the new $k$ centroids
      end
  end
return ($clustering\_solution$)
```

Figure 1.3: The K-MEANS procedure.

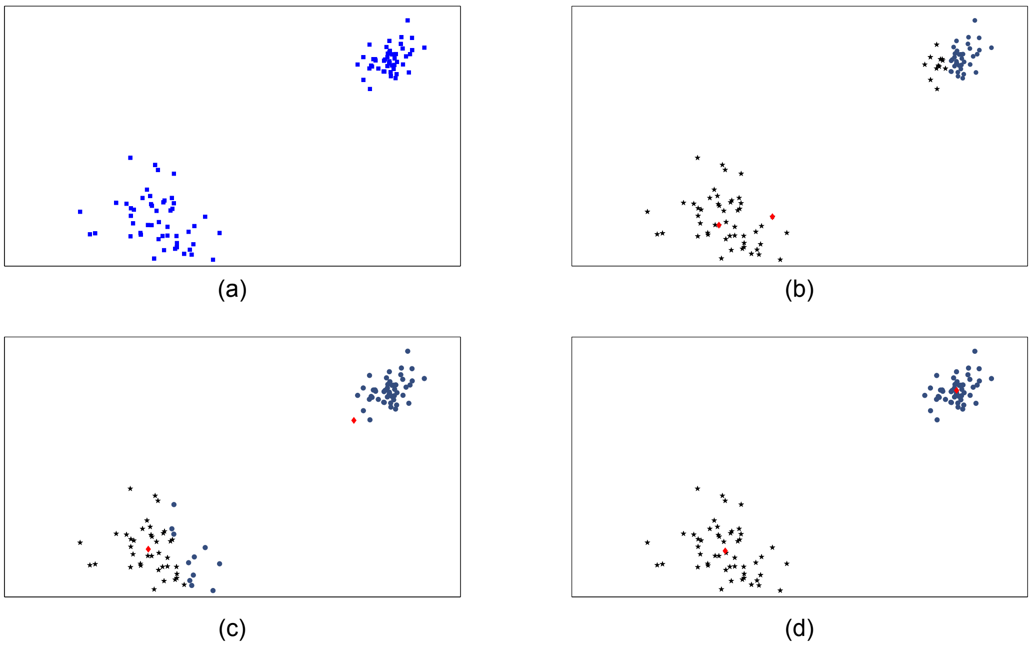

Figure 1.4: (a) Dataset. (b) Cluster membership after the first step. (c) Cluster membership after the second step. (d) Cluster membership after the last step.

In what follows, the three problem statements in which that question can be cast [87] are detailed.

Let C_j be a reference classification for Σ consisting of j classes. That is, C_j may either be a partition of Σ into j groups, usually referred to as the *gold standard*, or a division of the universe generating Σ into j categories, usually referred to as *class labels*. An *external index* E is a function that takes as input a reference classification C_j for Σ and a partition P_k of Σ and returns a value assessing how close the partition is to the reference classification. It is external because the quality assessment of the partition is established via criteria external to the data, i.e., the reference classification. Notice that it is not required that $j = k$. An *internal index* I is a function defined on the set of all possible partitions of Σ and with values in $\mathbb{R}$. It should measure the quality of a partition according to some suitable criteria. It is internal because the quality of the partition is measured according to information contained in the dataset without resorting to external knowledge. The first two problems are:

(**Q.1**) Given C_j, P_k and E, measure how far is P_k from C_j, according to E.

(**Q.2**) Given P_k and I, establish whether the value of I computed on P_k is unusual and therefore surprising. That is, significantly small or significantly large.

Notice that the two questions above try to assess the quality of a clustering solution P_k consisting of k groups, but they give no indication on what the "right number" of clusters is. In order to get such an indication, one is interested in the following:

(**Q.3**) Given: (Q.3.a) A sequence of clustering solutions $P_1, \ldots, P_s$, obtained for instance via repeated application of a clustering algorithm A; (Q.3.b) a function R, usually referred to as a *relative index*, that estimates the relative merits of a set of clustering solutions. One is interested in identifying the partition P_{k^*} among the ones given in (Q.3.a) providing the best value of R. In what follows, the optimal number of clusters according to R is referred to as k^*.

The clustering literature is extremely rich in mathematical functions suited for the three problems outlined above [73]. The crux of the matter is to establish quantitatively the threshold values allowing one to say that the value of an index is significant enough. That naturally leads to briefly mention hypothesis testing in statistics, from which one can derive procedures to assess the statistical significance of an index. As will be evident in the following sections, those procedures are rarely applied in microarray data analysis, being preferred to less resource-demanding heuristics that are validated experimentally.

1.4 Cluster Significance for a Given Statistic: a General Procedure

A *statistic* T is a function of the data capturing useful information about it, i.e., it can be one of the indices mentioned earlier. In mathematical terms, it is a random variable and its distribution describes the relative frequency with which values of T occur, according to some assumptions. In turn, since T is a random variable, one implicitly assumes the existence of a background or reference probability distribution for its values. That implies the existence of a sample space. A *hypothesis* is a statement about the frequency of events in the sample space. It is tested by observing a value of T and by deciding how unusual it is, according to the probability distribution one is assuming for the sample space. In what follows, one assumes that the higher the value of T, the more unusual it is, the symmetric case being dealt with similarly.

The most common hypothesis tested for in clustering is the *null hypothesis* H_0: there is no structure in the data (i.e. $k = 1$). Testing for H_0 with a statistic T in a dataset D means to compute T on D and then decide whether to reject or not to reject H_0. In order to decide, one needs to establish how significant is the value found with respect to a background probability distribution of the statistic T under H_0. That means one has to formalize the concept of "no structure" or "randomness" in the data. Among the many possible ways, generally referred to as *null models*, the most relevant proposed in the clustering literature [25, 68, 87, 149] are introduced, together with an identification of which one is well suited for microarray data analysis [48, 164]:

Unimodality Hypothesis. A new dataset D' is generated as follows: the variables describing the items are randomly selected from a unimodal distribution (e.g. normal). This null model typically is not applied to microarray data, since it gives a high probability of rejection of the null hypothesis. For instance, that happens when the data are sampled from a distribution with a lower kurtosis than the normal distribution, such as the uniform distribution [149]. Fig. 1.5(a) reports an example of a dataset generated via the unimodality hypothesis.

Random Graph Hypothesis. The entries of the dissimilarity/distance matrix S are random. That is, one assumes that, in terms of a linear order relation capturing proximity, all the entries of the lower triangular part of S are equally likely, i.e., $S_{i,j} = \frac{1}{[n(n-1)/2]!}$ for $1 \leq i \leq n$ and $1 \leq j \leq i$. This null model is not applied to microarray data, since it does not preserve the distances that may be present among items.

Random Label Hypothesis. All permutations of the items are equally likely with respect to some characteristic, such as a *priori* class membership. In order to use this model, one needs to specify the a priori classification of the data. Each permutation has a probability $\frac{1}{n!}$. In particular, for microarray data, it coincides with the so called *Permutational Model*, detailed in what follows.

- *Permutational Model* (`Pr` for short), it generates a random data matrix by randomly permuting the elements within the rows and/or the columns of D. Some variants of this model have been studied for binary pattern matrices [76, 160, 168]. In order to properly implement this model, care must be taken in specifying a proper permutation for the data, since some similarity and distance functions are insensitive to permutations of coordinates. That is, although D' is a random permutation of D, it may happen that the distance or similarity among the points in D' is the same as in D, resulting in indistinguishable datasets for clustering algorithms. This latter model may not be suitable for microarray data with very small sample sizes (conditions), since one will not obtain enough observations (data points) to estimate the null model, even if one generates all possible permutations.

Random Position Hypothesis. The items can be represented by points that are randomly drawn from a region $\mathcal{R}$ in m-dimensional space. In order to use this model, one needs to specify the region within which the points have to be uniformly distributed. Two instances applied to microarray data [48, 62, 164] are distinguished:

- *Poisson Model* (`Ps` for short), where the region $\mathcal{R}$ is specified from the data. The simplest regions that have been considered are the m-dimensional hypercube and hypersphere enclosing the points specified by the matrix D [68]. Another possibility, in order to make the model more data-dependent, is to choose the convex hull enclosing the points specified by D. Fig. 1.5(b) reports an example of a dataset D' generated by `Ps`, where the region $\mathcal{R}$ (the box in red) is obtained from the dataset D reported in Fig.1.4(a).

- *Poisson Model Aligned with Principal Components of the Data* (`Pc` for short), where Tibshirani et al. [164], following Sarle [149], propose to align the region $\mathcal{R}$ with the principal components of the data matrix D. In detail, assuming that the columns of D have mean zero, let $D = UXV^T$ be its singular value decomposition . Let $\widehat{D} = DV$. One uses $\widehat{D}$ as in `Ps` to obtain a dataset $\widehat{D}'$. Then one back transforms via $D' = \widehat{D}'V^T$ to obtain the new dataset. Fig. 1.5(c) reports an example of a dataset D' generated by `Pc`, where the region $\mathcal{R}$ (the box in red) is obtained from the dataset D reported in Fig.1.4(a).

It is worth pointing out that in a multivariate situation, one is not able to choose a generally applicable and useful reference distribution: the geometry of the particular null distribution matters [149, 164]. Therefore, in two or more dimensions, and depending on the test statistic, the results can be very sensitive to the region of support of the reference distribution [48, 149].

Once a null model has been agreed upon, one would like to obtain formulas giving the value of T under the null model and for a specific set of parameters. Unfortunately, not too many such formulae are available. In fact, in most cases, one needs to resort to a Monte Carlo simulation applied to the context of assessing the significance of a partition of the

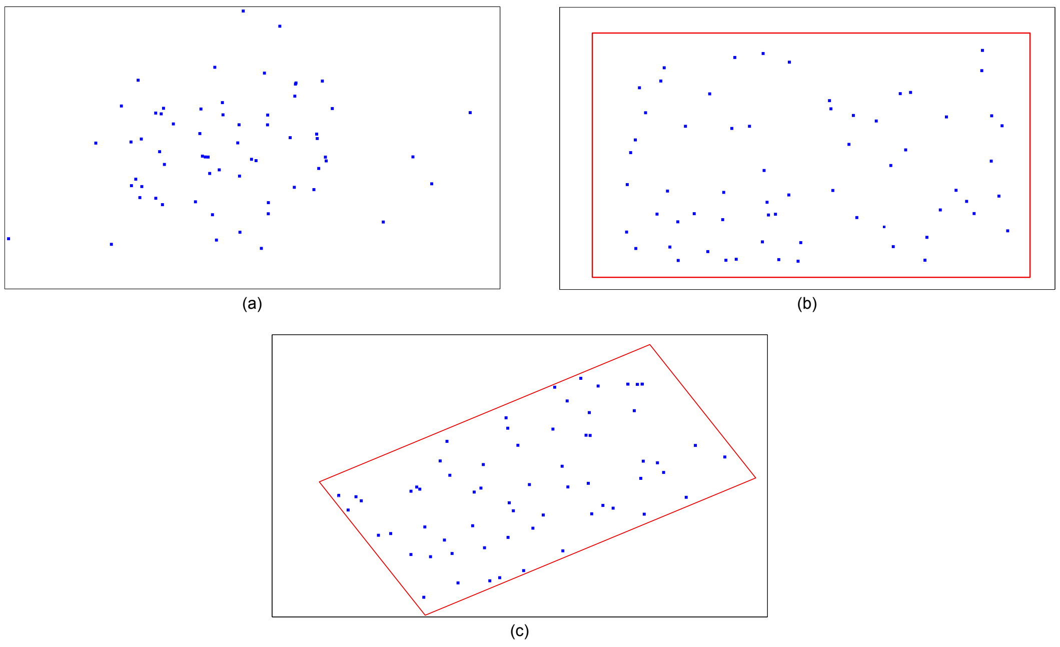

Figure 1.5: Dataset generated via (a) the Unimodality Hypothesis; (b) the Poisson Null Model; (c) the Poisson Null Model Aligned with Principal Components of the Data.

data into k clusters. In technical terms, it is a p-value test assessing whether in the dataset there exist k clusters, based on T and the null model for H_0. It is referred to as MECCA, an abbreviation for Monte Carlo Confidence Analysis, and it is described in Fig. 1.6. The procedure is also a somewhat more general version of a significance test proposed and studied by Gordon [67, 68] for the same problem. It takes as input an integer ℓ (the number of iterations in the Monte Carlo simulation), a clustering algorithm A, a dataset D, the function T, a partition P_k of D obtained via algorithm A and a parameter $\alpha \in [0, 1]$ indicating the level of "significance" for the rejection of H_0. It returns a value $p \in [0, 1]$. If $p < \alpha$, the null hypothesis of no cluster structure in the data is to be rejected at significance level α. Else, it cannot be rejected at that significance level.

A few remarks are in order. As pointed out by Gordon, significance tests aiming at assessing how reliable is a clustering solution are usually not carried out in data analysis. Microarrays are no exception, although sophisticated statistical techniques specific for those data have been designed (e.g. [49]). One of the reasons is certainly their high computational demand. Another, more subtle, reason is that researchers expect that "some structure" is present in their data. Nevertheless, a general procedure, like MECCA, is quite useful as a paradigm illustrating how one tries to assess cluster quality via a null model and a statistic T. Indeed, one computes the observed value of T (on the real data). Then, one computes, via a Monte Carlo simulation, enough values of T, as expected from the formalization of H_0 via the null model. Finally, one checks how "unusual" is the value of the observed statistic with respect to its expected value, as estimated by a Monte Carlo simulation. In Chapters 2 and 3, two methods that one describes in this thesis resort to the same principles and guidelines of MECCA, although they are more specific about the statistic that is relevant in order to identify the number of clusters in a dataset.

```
MECCA($\ell, A, D, T, P_k, \alpha$)

 for $i \leftarrow 1$ to $\ell$ do
   begin
 1.  Compute a new data matrix $D_i$, by using the chosen null model.
 2.  Partition $D_i$ into a set of $k$ clusters $P_{i,k}$ by using the algorithm $A$.
   end
 for $i \leftarrow 1$ to $\ell$ do
   begin
 3.  Compute $T$ on $P_{i,k}$.
 4.  Let $SL$ be the non-decreasing sorted array of $T$ values.
   end
 5. Let $V$ denote the value of $T$ computed on $P_k$.
 6. Let $p$ be the proportion of the values in $SL$ larger than $V$.
 return ($p$)
```

Figure 1.6: The MECCA procedure.

1.5 Data Generation/Perturbation Techniques

The null models described in Section 1.4 can be seen as an instance of a very general procedure, in what follows referred to as DGP, that generates new datasets from a given one. Such a procedure takes as input a dataset D, of size $n \times m$, together with other parameters and returns a new dataset D' of size $n' \times m'$, with $n' \leq n$ and $m' \leq m$. In this section, additional instances of DGP are detailed. They are used in microarray data analysis to generate data points in order to compute a cluster quality measure. Each of them can be thought of as a paradigm in itself and therefore in this section only an outline is provided.

1.5.1 Subsampling

The simplest way to generate a new dataset D' from D is to take random samples from it. Although simple, this approach critically depends on whether the sampling is performed without or with replacement. The first type of method is referred to as *subsampling*. It is widely used in clustering and briefly discussed here. The second method is referred to as *bootstrapping* and, although fundamental in statistics [50], it is hardly used in cluster validation as pointed out and discussed in [87, 122].

Formally, a subsampling procedure takes as input a dataset D and a parameter β, with $0 < \beta < 1$, and gives as output a percentage β of D, i.e., the dataset D' has size $n' \times m$, with $n' = \lceil \beta n \rceil$. D' is obtained via the extraction of n' items (i.e. rows) from D, which are usually selected uniformly and at random.

The aim of subsampling procedures is to generate a reduced dataset D' that captures the structure (i.e. the right number of clusters) in the original data. Intuitively, both the chances to achieve that goal and the time required by procedures using D' increase with β. In order to have a good trade-off between the representativeness of D' and the speed of the methods using it, a value of $\beta \in [0.6, 0.9]$ is used in the literature (e.g. [16, 48, 63, 122]).

The subsamping technique does not guarantee that each cluster of D is represented in D', i.e., the random extraction could not select any elements of a given cluster. Therefore, Hansen et al. [74] propose a heuristic referred to as *proportionate stratified sampling* as an alternative that may take care of the mentioned problem. In that case, D' is generated first by clustering D and then by selecting a given percentage β of the elements in each cluster. Proportionate stratified sampling gives no formal guarantee that the entire cluster structure of D is present in D'.

1.5.2 Noise Injection

Noise injection is a widely applied perturbation methodology in computer science (see [17, 24, 95, 117, 143, 175] and reference therein). However, it is not widely applied in clustering. The main idea is to generate D' by adding a random value, i.e., a "perturbation", to each of the elements of D. Perturbations are generated via some random process, i.e., a probability distribution whose parameters can be directly estimated from D. In a study about melanoma patients, Bittner et al. [24] propose to perturb the original dataset by adding Gaussian noise to its elements in order to assess cluster stability. Following up, Wolfinger et al. [175] report that perturbing the data via a Gaussian distribution provides good stability results for several microarray datasets. As for parameter estimation, McShane et al. [117] propose to compute the variance of experiments in each row of D and then to use the median of the observed distribution as the variance in the Gaussian distribution.

1.5.3 Dimensionality Reduction Methods

The methodology described in this section is, in most cases, the dual of subsampling, since the main idea is to obtain D' by reducing the number of columns of D while trying to preserve its cluster structure. Since each element, i.e. row, of D is a point in m-dimensional space, one has a dimensionality reduction in the data.

A well established dimensionality reduction method in data analysis is the *Principal Component Analysis* (PCA for short) [87]. Although it is a standard method for dimensionality reduction from a statistical point of view, it is not used in conjunction with stability-based internal validation measures (detailed in Chapter 3) because of its determinism in generating D'. Note, however, that the main idea of principal components is used in conjunction with null models (see for example the (M.2) model described in Section 1.4).

The following three techniques of dimensionality reduction seem to be of use in this area. The first one is rather trivial since it consists of randomly selecting the columns of D (cf. [156]). However, the cluster structure of D is unlikely to be preserved and this approach may introduce large distortions into gene expression data, which then result in the introduction of biases into stability indices (detailed in Chapter 3), as reported in [20]. More sensible approaches for dimensionality reduction are Non-negative Matrix Factorization (NMF for short) and randomized dimensionality reduction techniques reported in the following. This latter is discussed next, while discussion of the former is given in Chapter 4, while an exhaustive benchmarking of NMF as a clustering algorithm is provided in Chapter 5.

1.5.3.1 Randomized Dimensionality Reduction

The technique consists of the use of a family of transformations that try to preserve the distance with an "ε distortion level" between the elements of D. Intuitively, if two elements are "close" in D, according to some distance function d, they should be "close" in D'. Let f be a transformation from D in D', $f(\sigma_i)$ and $f(\sigma_j)$ be the projections of two elements σ_i and σ_j of D into D'. Let

$$d_f(\sigma_i, \sigma_j) = \frac{d(\sigma_i, \sigma_j)}{d(f(\sigma_i), f(\sigma_j))}.$$

If $d_f = 1$ the distance of the two elements is preserved. When $1 - \varepsilon \leq d_f(\sigma_i, \sigma_j) \leq 1 + \varepsilon$, one says that the function f preserves the distance with an "ε distortion level". The Johnson-Lindenstrauss Lemma and *random projections* are the keys to all the randomized dimensionality reduction based techniques. Intuitively, for a fixed distortion level ε, the Johnson-Lindenstrauss Lemma gives nearly optimal bounds to the value of m' (cf. [9]), formally:

Lemma 1 (Johnson-Lindenstrauss [89]). *For any $0 < \varepsilon < 1$ and any integer n, let m' be a positive integer such that*

$$m' > 4(\varepsilon^2/2 - \varepsilon^2/3)^{-1} \log n.$$

For any set V of n points in $\mathbb{R}^m$ there is a map function $f : \mathbb{R}^m \to \mathbb{R}^{m'}$ such that for all $u, v \in V$

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2$$

The interested reader will find two independent simplified versions of the proof of the above Lemma in [39, 85] and extensions to other spaces and distances in [6, 22, 37, 90]. It is possible to determine a function f that satisfies the Lemma with high probability, e.g., at least $2/3$, in randomized polynomial time [39, 85].

Since the projection into the new smaller space is a time consuming task, several heuristics have been proposed in the literature. Some of them are based on sparse projection matrices [5, 23], while a more innovative and recent approach has been proposed by Ailon and Chazelle [6] with the addition of the Fast Fourier Transform to the Johnson-Lindenstrauss Lemma. In conclusion, it is also worthy of mention that the dimensionality reduction techniques described here are tightly connected to problems as approximate nearest neighbor [12, 85] of relevance also for clustering.

CHAPTER 2

Fundamental Validation Indices

In this chapter, some basic validation techniques are presented. In detail, external and internal indices are outlined. In the scholarly literature, the terms index is also referred to as measure. Following the literature, in this dissertation both nomenclatures are used. The external and internal indices differ fundamentally in their aims, and find application in distinct experimental settings. In particular, three external indices that assess the agreement between two partitions are presented. Moreover, four internal measures useful to estimate the correct number of clusters present in a dataset, based on: compactness, hypothesis testing in statistics and jackknife techniques are also discussed. One of the topics of this thesis is the study of a relevant paradigm of internal validation measure based on the notion of cluster stability. For this reason this paradigm and the relative instances are thoroughly discussed in Chapter 3.

2.1 External Indices

In this section three external indices, namely formulae, are defined. Such measures establish the level of agreement between two partitions. Usually, for a given dataset, one of the partitions is a reference classification of the data while the other one is provided as output by a clustering algorithm.

Let $C = \{c_1, \ldots, c_r\}$ be a partition of the items in Σ into r classes and $P = \{p_1, \ldots, p_t\}$ be another partition of Σ into t clusters. With the notation of Section 1.3, C is an external partition of the items, derived from the reference classification, while P is a partition obtained by some clustering method. Let $n_{i,j}$ be the number of items in both c_i and p_j, $1 \leq i \leq r$ and $1 \leq j \leq t$. Moreover, let $|c_i| = n_{i.}$ and $|p_j| = n_{.j}$. Those values can be conveniently arranged in a *contingency table* (see Table 2.1).

2.1.1 Adjusted Rand Index

Let a be the number of pairs of items that are placed in the same class in C and in the same cluster in P; let b be the number of pairs of items placed in the same class in C but not in the same class in P; let c be the number of pairs of items in the same cluster in P but not in the same cluster in C; let d be the number of pairs of items in different classes and different clusters in both partitions. The information needed to compute a, b, c and d can be derived from Table 2.1. One has:

$$a = \sum_{i,j} \binom{n_{i,j}}{2}, \tag{2.1}$$

$$b = \sum_{i} \binom{n_{i.}}{2} - a, \tag{2.2}$$

$$c = \sum_{j} \binom{n_{.j}}{2} - a. \tag{2.3}$$

Class \ *Cluster*	p_1	p_2	$\ldots$	p_t	Sums
c_1	$n_{1,1}$	$n_{1,2}$	$\ldots$	$n_{1,t}$	$n_{1.}$
c_2	$n_{2,1}$	$n_{2,2}$	$\ldots$	$n_{2,t}$	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
c_r	$n_{r,1}$	$n_{r,2}$	$\ldots$	$n_{r,t}$	$n_{r.}$
Sums	$n_{.1}$	$n_{.2}$	$\ldots$	$n_{.t}$	$n_{..} = n$

Table 2.1: Contingency table for comparing two partitions

Moreover, since $a + b + c + d = \binom{n}{2}$, one has:

$$d = \binom{n}{2} - (a + b + c). \tag{2.4}$$

Based on those quantities, the `Rand` index R is defined as [142]:

$$R = \frac{a + d}{a + b + c + d} \tag{2.5}$$

Notice that, since $a + d$ is the number of pairs of items in which there is agreement between the two partitions, R is an index of agreement of the two partitions with value in $[0, 1]$. The main problem with R is that its value on two partitions picked at random does not take a constant value, say zero. So, it is difficult to establish, given two partitions, how significant (distant from randomness) is the concordance between the two partitions, as measured by the value of R. In general, given an index, it would be appropriate to take an adjusted version ensuring that its expected value is zero when the partitions are selected at random and one when they are identical. That can be done according to the following general scheme:

$$\frac{\textit{index} - \textit{expected index}}{\textit{maximum index} - \textit{expected index}} \tag{2.6}$$

where *maximum index* is the maximum value of the *index* and *expected index* is its expected value derived under a suitably chosen model of random agreement between two partitions, i.e. the null hypothesis. The `Adjusted Rand` Index R_A is derived from (2.5) and (2.6) using the generalized hypergeometric distribution as the null hypothesis. That is, it is assumed that the row and column sums in Table 2.1 are fixed, but the two partitions are picked at random. One has [84]:

$$R_A = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - \frac{\left[\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right]}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \frac{\left[\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right]}{\binom{n}{2}}}$$

R_A has a maximum value of one, when there is a perfect agreement between the two partitions, while its expected value of zero indicates a level of agreement due to chance. Moreover, R_A can take on a larger range of values with respect to R and, in particular,

$Class \setminus Cluster$	p_1	p_2	p_3	p_4	v_5	Sums
c_1	1	4	2	1	2	10
c_2	0	1	1	0	1	3
c_3	1	2	0	2	0	5
c_4	2	1	0	1	2	6
c_5	1	0	1	0	3	5
Sums	5	8	4	4	8	$n = 29$

Table 2.2: Contingency table example

may be negative [180]. Therefore, the two partitions are in significant agreement if R_A assumes a non-negative value, substantially away from zero. Notice that R_A is a statistic on the level of agreement of two partitions of a dataset (see Section 1.4) while R is a simple indication of percentage agreement. To illustrate this point, consider two partitions of a set of 29 items giving rise to Table 2.2. Then $R = 0.677$, indicating a good percentage agreement while $R_A = -0.014$ and, being close to its expected value under the null model, it indicates a level of significance in the agreement close to the random case. In fact, the entries in the table have been picked at random. R_A is a statistic recommended in the classification literature [120] to compare the level of agreement of two partitions.

2.1.2 Fowlkes and Mallows Index

The `FM-index` [54] is also derived from the contingency Table 2.1, as follows:

$$FM_k = \frac{T_k}{\sqrt{U_k \cdot V_k}} \tag{2.7}$$

where:

$$T_k = \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij}^2 - n \tag{2.8}$$

$$U_k = \sum_{i=1}^{k} n_{i.}^2 - n \tag{2.9}$$

$$V_k = \sum_{j=1}^{k} n_{.j}^2 - n \tag{2.10}$$

The index has values in the range $[0, 1]$, with an interpretation of the values in that interval analogous to that provided for the values of R. An example can be obtained as in Section 2.1.1. Indeed, for Table 2.2, one has $FM_5 = 0.186$ indicating a low level of agreement between the two partitions.

2.1.3 The F-Index

The `F-index` [145] combines notions from information retrieval, such as precision and recall, in order to evaluate the agreement of a clustering solution P with respect to a reference partition C. Again, its definition can be derived from the contingency Table 2.1.

Given c_i and p_j, their relative precision is defined as the ratio of the number of elements of the class c_i within cluster p_j, and by the size of the cluster p_j. That is:

$$Prec(c_i, p_j) = \frac{n_{i,j}}{n_{\cdot j}} \tag{2.11}$$

Moreover, their relative recall is defined as the ratio of the number of elements of the class c_i within cluster p_j, divided by the size of the class c_i. That is:

$$Rec(c_i, p_j) = \frac{n_{i,j}}{n_{i\cdot}}. \tag{2.12}$$

The `F-index` is then defined as an harmonic mean that uses the precision and recall values, with weight b:

$$F(c_i, p_j) = \frac{(b^2+1) \cdot Prec(c_i, p_j) \cdot Rec(c_i, p_j)}{b^2 \cdot Prec(c_i, p_j) + Rec(c_i, p_j)}. \tag{2.13}$$

Equal weighting for precision and recall is obtained by setting $b = 1$. Finally, the overall `F-index` is:

$$F = \sum_{c_i \in C} \frac{n_{i\cdot}}{n} \cdot \max_{p_k \in P} F(c_i, p_k). \tag{2.14}$$

F is an index with value in the range $[0, 1]$, with an interpretation of the values in that interval analogous to that provided for the values of R. An example can be obtained as in Section 2.1.1. Indeed, for Table 2.2, one has $F = 0.414$, indicating low level of agreement between the two partitions.

2.2 Internal and Relative Indices

Internal indices should assess the merits of a partition, without any use of external information. Then, a Monte Carlo simulation can establish if the value of such an index on the given partition is unusual enough for the user to gain confidence that the partition is good. Unfortunately, this methodology is rarely used in data analysis for microarrays, as stated in [73]. Internal indices are also a fundamental building block in order to obtain relative indices that help to select, among a given set of partitions, the "best" one.

In this section, four relative indices are presented, starting with the ones based on compactness. Then methods that are based on hypothesis testing and the jackknife approach are presented.

2.2.1 Methods Based on Compactness

The measures presented here assess cluster compactness. The most popular compactness measures are based on the sum-of-squares. In what follows, two of the prominent measures in that class are detailed.

2.2.1.1 Within Cluster Sum-of-Squares

An internal measure that gives an assessment of the level of compactness of each cluster in a clustering solution is the Within Cluster Sum of Squares (`WCSS` for short). Let $\mathcal{C} = \{c_1, \ldots, c_k\}$ be a clustering solution, with k clusters. Formally, let

$$D_r = \sum_{j \in c_r} ||\sigma_j - \overline{\sigma_r}||^2 \tag{2.15}$$

where $\overline{\sigma_r}$ is the centroid of cluster c_r. Then, one has:

$$\texttt{WCSS}(k) = \sum_{r=1}^{k} D_r. \tag{2.16}$$

By analyzing the behavior of WCSS in $[1, k_{max}]$, as a function of k, one can estimate the correct number of cluster k^* in the dataset. Intuitively, for values $k < k^*$, the compactness of each cluster should substantially increase, causing a substantial decrease in WCSS. In other words, one should observe in the WCSS curve a decreasing marginal improvement in terms of cluster compactness after the value k^*. The following heuristic approach comes out [79]: Plot the values of WCSS, computed on the given clustering solutions, in the range $[1, k_{max}]$; choose as k^* the abscissa closest to the "knee" in the WCSS curve. Fig. 2.1 provides an example of the WCSS curve computed on the dataset of Fig. 1.4(a) with K-means-R (see Section 1.2.2) for $k \in [1, 10]$. Indeed, the dataset has two natural clusters and the plot of the WCSS curve in Fig. 2.1 indicates $k^* = 2$. As it will be clear in Section 6.2.1, the prediction of k^* with WCSS is not so easy on real datasets, since the behavior of WCSS is not so regular as one expects.

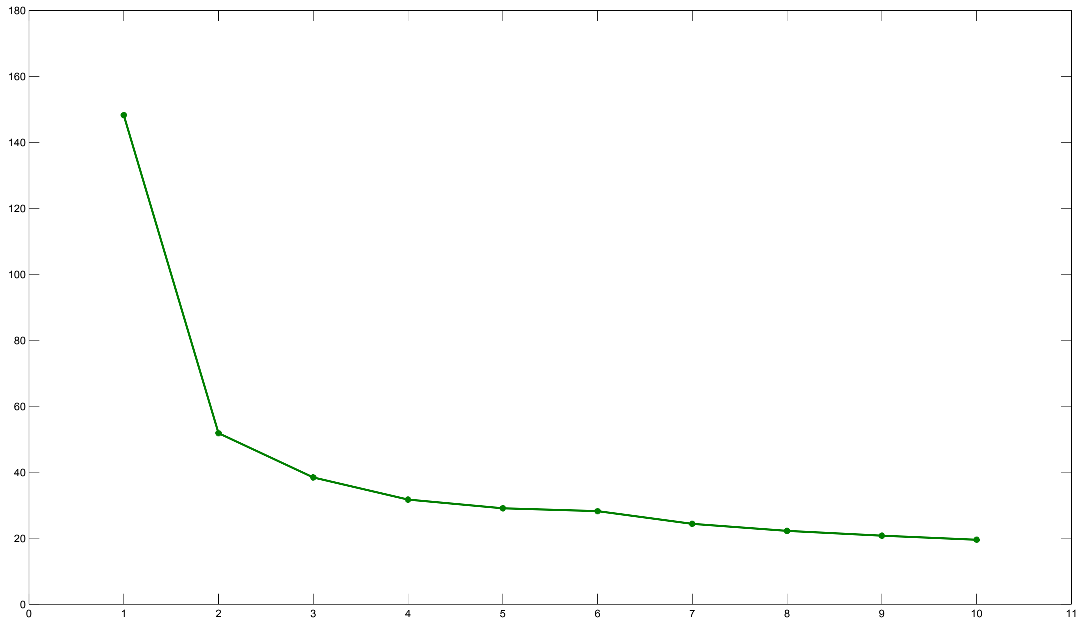

Figure 2.1: Plot of the values of WCSS.

2.2.1.2 Krzanowski and Lai Index

By elaborating on an earlier proposal by Marriot [114], Krzanowski and Lai [101] proposed an internal measure, which is referred to as KL. It is based on WCSS, but it is automatic, i.e., a numeric value for k^* is returned. Let

$$DIFF(k) = (k-1)^{2/m}\texttt{WCSS}(k-1) - k^{2/m}\texttt{WCSS}(k). \tag{2.17}$$

with $2 \leq k \leq k_{max}$.

Recall from Section 2.2.1.1 the behavior of WCSS, with respect to k^*. Based of those considerations, one expects the following behavior for $DIFF(k)$:

(i) for $k < k^*$, both $DIFF(k)$ and $DIFF(k+1)$ should be large positive values.

(ii) for $k > k^*$, both $DIFF(k)$ and $DIFF(k+1)$ should be small values, and one or both might be negative.

(iii) for $k = k^*$, $DIFF(k)$ should be large positive, but $DIFF(k+1)$ should be relatively small (might be negative).

Based on these considerations, Krzanowski and Lai propose to choose the estimate on the number of clusters as the k maximizing:

$$KL(k) = \left| \frac{DIFF(k)}{DIFF(k+1)} \right|. \tag{2.18}$$

That is,

$$k^* = \underset{2 \leq k \leq k_{max}}{\arg\max}\ KL(k). \tag{2.19}$$

Notice that $KL(k)$ is not defined for the important special case of $k = 1$, i.e., there is no cluster structure in the data. Figure 2.2(a) reports an example of $DIFF(k)$ computation with K-means-R (see Section 1.2.2) on the dataset of Fig. 1.4(a); the corresponding KL values are reported in Fig.2.2(b). Notice that the KL curve has a local maximum on $k = 3$ (value close to the correct number of classes) but based on (2.19) the prediction is $k^* = 5$.

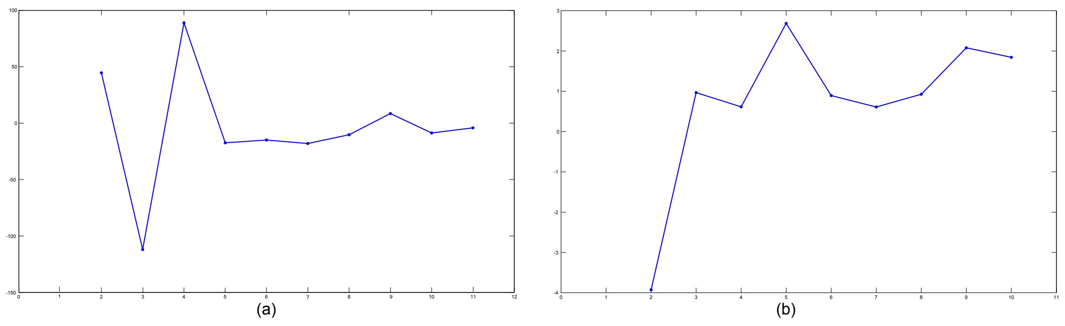

Figure 2.2: (a) Plot of the values of $DIFF$. (b) Plot of the values of KL.

2.2.2 Methods Based on Hypothesis Testing in Statistics

The measures presented so far are either useless or not defined for the important special case $k = 1$. In this thesis, two methods based on hypothesis testing proposed by Dudoit and Fridlyand [48] and Tibshirani et al. [164] are considered. The former is a clever combination of the MECCA hypothesis testing paradigm (see Section 1.4) and stability techniques, and for this reason is detailed in Chapter 3.

Tibshirani et al. [164] brilliantly combine the ideas of Section 1.4 with the WCSS heuristic, to obtain an index that can deal also with the case $k = 1$. It is referred to as the Gap

Statistics and, for brevity, it is denoted as `Gap`.
The intuition behind the method is brilliantly elegant. Recall, from the previous subsection that the "knee" in the `WCSS` curve can be used to predict the real number of cluster in the dataset. Unfortunately, the localization of such a value may be subjective. Consider the curves in Fig. 2.3. The curve in green at the bottom of the figure is the `WCSS` given in Fig. 2.1. The curve in red at the top of the figure is the *average* `WCSS`, computed on ten datasets generated from the original data via the `Ps` null model. As it is evident from the figure, the curve on the top has a nearly constant slope: an expected behavior on datasets with no cluster structure in them. The vertical lines indicate the gap between the null model curves and the curve computed by K-means-R, which supposedly captures "cluster structure" in the dataset. Since `WCSS` is expected to decrease sharply up to k^*, on the real dataset, and it has a nearly constant slope on the null model datasets, the length of the vertical segments is expected to increase up to k^* and then to decrease. In fact, in the figure, if one takes as the prediction for k^* the first local maximum of the gap values (data not shown), one has $k^* = 2$, the correct number of classes in the dataset. Normalizing the `WCSS` curves via logs and accounting also for the simulation error, such an intuition can be given under the form of a procedure in Fig. 2.4, which is strikingly similar to MECCA, as discussed shortly (see Section 1.4). The first three parameters are as in that procedure, while the last one states that the search for k^* must be done in the interval $[1, k_{max}]$.

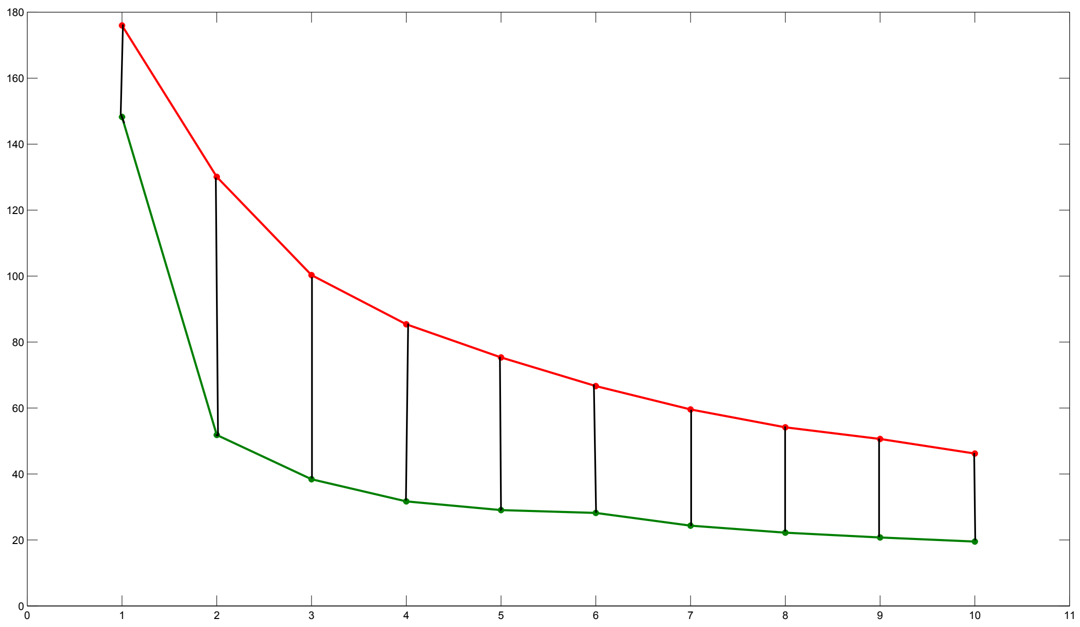

Figure 2.3: A geometric interpretation of the Gap Statistics.

Now, $\log(\texttt{WCSS}(k))$ is the statistic T used to assess how reliable is a clustering solution with k clusters. The value of that statistic is computed on both the observed data and on data generated by the chosen null model. Then, rather than returning a p-value, the procedure returns the first k for which "the gap" between the observed and the expected statistic is at a local maximum. With reference to step 7 of procedure GP (see Fig. 2.4), it is worth pointing out that the adjustment due to the $s(k + 1)$ term is a heuristic meant to account for the Monte Carlo simulation error in the estimation of the expected value of

```
GP($\ell, A, D, k_{max}$)

for $i \leftarrow 1$ to $\ell$ do
1. Compute a new data matrix $D_i$, using the chosen null model.
      Let $D_0$ denote the original data matrix.
for $i \leftarrow 1$ to $\ell$ do
  begin
    for $k \leftarrow 1$ to $k_{max}$ do
2.    Compute a clustering solution $P_{i,k}$ on $D_i$ using algorithm $A$.
  end
for $i \leftarrow 1$ to $\ell$ do
  begin
    for $k \leftarrow 1$ to $k_{max}$ do
3.    Compute $\log(\texttt{WCSS}(k))$ on $P_{i,k}$ and store the result in matrix $SL[i,k]$.
  end
for $k \leftarrow 1$ to $k_{max}$ do
  begin
4.    $\texttt{Gap}(k) \leftarrow \frac{1}{\ell}\sum_{i=1}^{\ell} SL[i,k] - SL[0,k]$.
5.    Compute the standard deviation $sd(k)$ of the set of numbers
      $\{SL[1,k], \ldots, SL[\ell,k]\}$
6.    $s(k) \leftarrow \left(\sqrt{1+\frac{1}{\ell}}\right) sd(k)$.
  end
7. $k^*$ is the first value of $k$ such that $\texttt{Gap}(k) \geq \texttt{Gap}(k+1) - s(k+1)$.
return $(k^*)$
```

Figure 2.4: The Gap Statistics procedure.

$\log(\texttt{WCSS}(k))$ (cf. [164]).

As discussed in Chapter 6, the prediction of k^* is based on running a certain number of times the procedure GP. Then one takes the most frequent outcome as the prediction. It is worth pointing out that further improvements and generalizations of `Gap` have been proposed in [178].

2.2.3 Methods Based on Jackknife Techniques: FOM

Figure of Merit (`FOM` for short) is a family of internal validation measures introduced by Yeung et al. [181], specifically for microarray data. Such a family is based on the jackknife approach and it has been designed for use as a relative index assessing the predictive power of a clustering algorithm, i.e., its ability to predict the correct number of clusters in a dataset. It has also been extended in several directions by Datta and Datta [40]. Experiments by Yeung et al. show that the `FOM` family of measures satisfies the following properties, with a good degree of accuracy. For a given clustering algorithm, it has a low value in correspondence with the number of clusters that are really present in the data. Moreover, when comparing clustering algorithms for a given number of clusters k, the lower the value of `FOM` for a given algorithm, the better its predictive power. In what follow, a review of this work is given, using the `2-norm FOM`, which is the most used instance in the `FOM` family.

Assume that a clustering algorithm is given the data matrix D with column e excluded. Assume also that, with that reduced dataset, the algorithm produces k clusters $c_1, \ldots, c_k$. Let $D(\sigma, e)$ be the expression level of gene σ and $m_i(e)$ be the average expression level of condition e for genes in cluster c_i. The `2-norm FOM` with respect to k clusters and condition e is defined as:

$$\texttt{FOM}(e,k) = \sqrt{\frac{1}{n}\sum_{i=1}^{k}\sum_{x\in c_i}(D(x,e) - m_i(e))^2}. \tag{2.20}$$

Notice that $\texttt{FOM}(e,k)$ is essentially a root mean square deviation. The `aggregate 2-norm FOM` for k clusters is then:

$$\texttt{FOM}(k) = \sum_{e=1}^{m}\texttt{FOM}(e,k). \tag{2.21}$$

Both formulae (2.20) and (2.21) can be used to measure the predictive power of an algorithm. The first one offers more flexibility, since one can pick any condition, while the second one offers a total estimate over all conditions. So far, (2.21) is the formula used the most in the literature. Moreover, since the experimental studies conducted by Yeung et al. show that $\texttt{FOM}(k)$ behaves as a decreasing function of k, an adjustment factor has been introduced to properly compare clustering solutions with different numbers of clusters. A theoretical analysis by Yeung et al. provides the following adjustment factor:

$$\sqrt{\frac{n-k}{n}}. \tag{2.22}$$

When (2.22) divides (2.20), (2.20) and (2.21) are referred to as *adjusted* `FOM`s. The adjusted aggregate `FOM` is used for the experiments in this thesis and, for brevity, it is referred to as `FOM`.

The use of `FOM` in order to establish how many clusters are present in the data follows the same heuristic methodology outlined for `WCSS`, i.e., one tries to identify the “knee” in the `FOM` plot as a function of the number of clusters. Fig. 2.5 provides an example, where the FOM curve is computed on the dataset of Fig. 1.4(a) with K-means-R. In this case, it is easy to see that the predicted value is $k^* = 6$, that is a value very far to the correct number of clusters in the dataset.

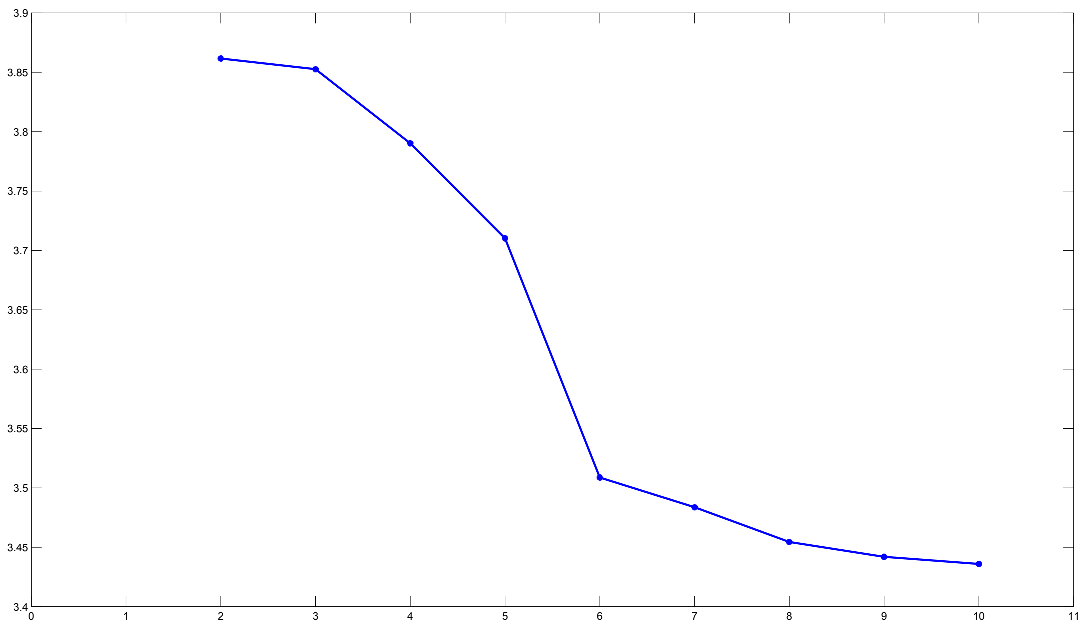

Figure 2.5: Plot of the values of FOM.

CHAPTER 3

The Stability Measure Paradigm and Its Instances

In this chapter, internal validation measures based on the notion of stability are presented. First, a general algorithmic paradigm is introduced, which can be seen as a generalization of earlier work by Breckenridge and Valentini. Then, it is shown that each of the known stability based measures is an instance of such a novel paradigm. Surprisingly, also `Gap` falls within the new paradigm.

3.1 An Intuitive Description

All the methods described in this chapter for model selection in clustering are related to the concept of *stability* which is now discussed in intuitive terms. A "good" algorithm should produce clustering solutions that do not vary much from one sample to another, when data points are repeatedly sampled and clustered. That is, the algorithm must be stable with respect to input randomization. Therefore, the main idea to validate a clustering solution is to use a measure the self-consistency of the data instead of using the classical concepts of isolation and compactness [73, 87].

The stability framework can be applied to problems (**Q.2**) and (**Q.3**), detailed in Section 1.3, and for convenience of the reader reported here again:

- (**Q.2**) compute a partition of D and assess the confidence of cluster assignments for individual samples;

- (**Q.3**) estimate the number of clusters, if any, in a dataset.

These two problems are strongly related, since it is possible to use the former problem to solve the latter.

In order to obtain a stability internal validation method, one needs to specify the following "ingredients":

1. a data generation/pertubation procedure;
2. a similarity measure between partitions;
3. a statistics on clustering stability;
4. rules on how to select the most reliable clustering(s).

Points 1 and 2 have been addressed in Sections 1.4-1.5 and 2.1, respectively. Points 3 and 4 are the main subject of study of this chapter. For problem (**Q.2**) and (**Q.3**) a first effort to formalize the steps and the ingredients of a solution based on "stability" are due to

Breckenridge [29] and Valentini [167], respectively. In particular, this latter formalization is done via a software library for the statistical computing environment R, which is referred to as `mosclust`. Indeed, this tool provides several macro operations in order to implement a stability internal measure. However, `mosclust` focuses its attention on similarity measures between only two partitions (see point 2) obtained from the same clustering algorithm and it does not provide any macro that takes into account a more general similarity measure.

Extensive experimental results (see [49, 62] and Chapter 6) show that this class of measures has an excellent predictive power. However there are some drawbacks and open problems associated with their use:

(a) As shown in [73], a given clustering may converge to a suboptimal solution owing to the shape of the data manifold and not to the real structure of the data. Thus, some bias in the stability indices are introduced.

(b) Ben-David et al. [15] show that stability methods based on resampling techniques, when cost-based clustering algorithms are used, may fail to estimate k^*, if the data is not symmetric.

(c) Stability methods have various parameters that a user needs to specify [62]. Those choices may affect both their time performance and their estimation of k^*.

Whit respect to the problem (b), it is unclear if these results may be extended to other stability based methods or to other more general classes of clustering algorithms. In this dissertation, one focuses on problem (c). Indeed, as it will be shown in Chapter 6, repeatedly generating and clustering data, i.e., the main cycle of the stability based methods, has a drastic influence on time performance. Therefore, design of fast approximation algorithms is needed in order to use these measures for large datasets. It is worth pointing out that in Chapter 7 an approximation scheme of the stability internal validation measures is proposed.

In this chapter a generalization of the efforts of Breckenridge and Valentini is proposed via two novel paradigms in order to solve the problems (**Q.2**) and (**Q.3**). The former is described in Section 3.2.1 and it is referred to as Stability Statistic. The latter is described in Section 3.2.2 is referred to as Stability Measure. Finally, in Section 3.3 several stability measure are presented as instances of the novel paradigm.

3.2 The Stability Statistic and the Stability Measure Paradigms

Recall from [87] that a *statistic* is a function of the data capturing useful information about it. A statistic assessing cluster stability is, intuitively, a measure of consistency of a clustering solution. In turn, information obtained from the statistic is used by the Stability Measure in order to estimate k^*. Since Stability Statistic is a “subroutine” of the Stability Measure paradigm, it is presented first. In what follows, a statistic is represented by a set S of records. For instance, in its simplest form, a statistic consists of a single real number, while in other cases of interest, it is a one or two-dimensional array of real numbers.

3.2.1 The Stability Statistic Paradigm

The paradigm for the collection of a statistic on cluster stability is best presented as a procedure, reported in Fig. 3.1. Its input parameters and macro operations are described in abstract form in Figs. 3.2 and 3.3, respectively, while its basic steps are described below.

```
Stability_Statistic($D_0, H, \alpha, \beta, < C_1, C_2, \dots, C_t >, k$)

$S^k = \emptyset$
while $H$ do
  begin
  1. $< D_1, D_2, \dots, D_l > \leftarrow < \text{DGP}(D_0, \beta), \text{DGP}(D_0, \beta), \dots, \text{DGP}(D_0, \beta) >$
  2. $< D_{T,0}, D_{T,1}, \dots, D_{T,l}, D_{L,0}, D_{L,1}, \dots, D_{L,l} > \leftarrow$ Split($< D_0, D_1, \dots, D_l >, \alpha$)
  3. $< G > \leftarrow$ Assign($< D_{T,0}, D_{T,1}, \dots, D_{T,l} >, < C_1, C_2, \dots, C_t >$)
  4. $< C_{i_1}, C_{i_2}, \dots, C_{i_q} > \leftarrow$ Train($< G >$)
  5. $< \hat{G} > \leftarrow$ Assign($< D_{L,0}, D_{L,1}, \dots, D_{L,l} >, < C_1, C_2, \dots, C_t >$)
  6. $< P_1, P_2, \dots, P_z > \leftarrow$ Cluster($\hat{G}, k$)
  7. $u \leftarrow$ Collect_Statistic($< P_1, P_2, \dots, P_z >$)
  8. $S^k \leftarrow S^k \bigcup \{u\}$
  end
return ($S^k$)
```

Figure 3.1: The Stability_Statistic procedure.

Input

- D_0: it is the input dataset.
- H: it is a test on the "adequacy" of a statistic S, i.e., it evaluates whether S contains enough information. Note that H could simply be a check of as to whether a given number c of iterations has been reached. In what follows, this simple test is denoted as $\hat{H}_c$.
- α: it is a number in the range $[0, 1]$.
- β: it is a sampling percentage, used by the DGP procedure (described in Section 1.5).
- $< C_1, C_2, \dots, C_t >$: it is a set of procedures, each of which is either a classifier or a clustering algorithm.
- k: it is the number of clusters in which a dataset has to be partitioned.

Figure 3.2: List of the input parameters used in the Stability_Statistic procedure.

MACRO OPERATIONS

- SPLIT: it takes as input a family of datasets $F_1, F_2, \ldots, F_w$ and a real number α in the range [0,1]. The procedure splits each F_i, $1 \leq i \leq w$, into two parts according to α, referred to as *learning* and *training* dataset and denoted with $F_{L,i}$ and $F_{T,i}$, respectively. That is, from each F_i, $\lceil \alpha n_i \rceil$ and $\lfloor (1-\alpha) n_i \rfloor$ rows are selected in order to obtain the corresponding $F_{T,i}$ and $F_{L,i}$, respectively, where n_i is the number of rows of F_i. Each $F_{T,i}$ and $F_{L,i}$ is given as output.

- ASSIGN: it takes as input a family of datasets and a set of procedures, each of which is either a classifier or a clustering algorithm. It returns a finite set of pairs in which the first element is a dataset and the second one is either a classifier or a clustering algorithm. Such an association is encoded via a bipartite graph G, where the datasets are represented by nodes in one partition and procedures in the other partition. Notice that the graph is not a matching, i.e., the same dataset can be assigned to different procedures and viceversa.

- TRAIN: it takes as input a set of pairs <dataset, classifier>, encoded as a bipartite graph, analogous to the one just discussed. For each pair, it gives as output the classifier trained with the corresponding dataset. Notice that the number q of trained classifiers returned as output is equal to the number of edges in the input graph.

- CLUSTER: it takes as input a set of pairs <dataset, classifier/clustering algorithm> and a positive integer k. Again, the set is encoded as a bipartite graph. For each pair, it gives as output a partition in k clusters obtained by the classifier/clustering algorithm on the corresponding input dataset. Notice that the number z of partitions returned as output is equal to the number of edges in the input graph.

- COLLECT_STATISTIC: it takes as input a set of partitions. It returns as output the statistic computed on the input set.

Figure 3.3: List of the macro operations used in the STABILITY_STATISTIC procedure.

A single iteration of the **while** loop is discussed. The loop is repeated until the condition H is satisfied, i.e., until enough information about the given statistic has been collected. In step 1, a set of perturbed datasets is generated from D_0 by a DGP procedure (see Section 1.5). In step 2, D_0 and all the datasets generated in the previous step are split in a learning and training dataset, according to the input parameter α. The next two steps train a subset of the classifiers on a subset of the training sets. In step 5, the bipartite graph $\hat{G}$ encodes the association between learning datasets and clustering procedures. In step 6, based on the association encoded by $\hat{G}$, the learning datasets are partitioned. Finally, in step 7, a statistic S^k is computed from those partitions and is given as output.

In the next subsection, some instances of this paradigm are discussed.

3.2.1.1 Instances

Here three incarnations of the Stability Statistic paradigm are provided. The first is *replicating analysis*, a ground-breaking method due to Breckenridge [29]. The other two are *BagClust1* and *BagClust2*, due to Dudoit and Fridlyand [49]. In all three cases, the procedures were proposed to improve a clustering solution for a fixed value of k (see problem (**Q.2**)), rather than to estimate the "true" number of clusters in D. However, as one will see in Section 3.3, *replicating analysis* and *BagClust2* play a key role in many internal stability methods.
The presentation of methods in this section is organized as follows: for each example, the input parameters setup is first described (see Fig. 3.2), then the Stability_Statistic is detailed.

- *Replicating analysis.*

    - *The input parameters setup*: β is not relevant and the simple test $\hat{H}_1$ is used to allow only one iteration of the **while** loop. Moreover, the set of procedures $< C_1, C_2, \ldots, C_t >$ has size two, i.e., it contains one classifier and one clustering algorithm, referred to as C_1 and C_2, respectively.

    - *The* Statistics_Stability *procedure*: step 1 is not performed. In step 2, the Split procedure is applied to D_0 only and it gives as output the training and learning dataset $D_{T,0}$ and $D_{L,0}$, respectively. Then, in steps 3-5, $D_{T,0}$ is used to train the classifier C_1. In steps 5 and 6, two partitions P_1 and P_2 of $D_{L,0}$ are produced, by C_1 and C_2, respectively. Finally, in step 7, the Collect_Statistic procedure measures the agreement between the two partitions P_1 and P_2 via an external index (see Section 2.1) in order to assess the stability structure of the dataset. For convenience of the reader the *replicating analysis* procedure is given in Fig. 3.4.

- *BagClust1.*

    - *The input parameters setup*: $\hat{H}_c$ is used as test, for a given number of iterations c. The set of procedures $< C_1, C_2, \ldots, C_t >$ consists only of one clustering algorithm, $\alpha = 0$ and $\beta = 1$. Moreover, each DGP is an instance of the same bootstrapping subsampling method (see Section 1.5.1). Since $\alpha = 0$, the Split procedure gives as output only the learning datasets, which are copies of the corresponding input dataset.

```
REPLICATING_ANALYSIS($D_0, \alpha, < C_1, C_2 >, k$)
 1. Split the input dataset in $D_L$ and $D_T$, the learning and training sets,
    respectively.
 2. Train the classifier $C_1$on $D_T$.
 3. Let $P_1$ and $P_2$ be the partitions of $D_L$
    into $k$ cluster with the use of $C_1$ and $C_2$, respectively.
 4. Let $e$ be the agreement measure between $P_1$ and $P_2$ obtained
    via an external index.
 return ($e$)
```

Figure 3.4: The *replicating analysis* procedure.

- *The* STATISTICS_STABILITY *procedure*: in step 1, a single DGP procedure is executed to generate D_1. Then, the SPLIT procedure takes as input D_0 and D_1 and it gives as output $D_{L,0} = D_0$ and $D_{L,1} = D_1$. In steps 5 and 6, the clustering procedure is applied to both D_0 and D_1 in order to obtain the partitions P_1 and P_2, respectively. The COLLECT_STATISTIC procedure permutes the elements assigned to the partition P_2 so that there is the maximum overlap with P_1. For each iteration of the **while** loop, the number of overlapping elements are counted and given as output of the method. From that statistic, a new partition is obtained by assigning each element of D_0 to a cluster via a majority vote system. That is, each element is assigned to the cluster for which it expressed the maximum of number of preferences. For convenience of the reader the*BagClust1* procedure is given in Fig. 3.5.

• *BagClust2.*

- *The input parameters setup*: as in *BagClust1.*
- *The* STATISTICS_STABILITY *procedure*: in step 1, a single DGP procedure is executed to generate D_1. Then, the SPLIT procedure takes as input D_0 and D_1 and it gives as output $D_{L,0} = D_0$ and $D_{L,1} = D_1$. In step 5, the bipartite graph $\hat{G}$ consists of only one node per partition, encoding the dataset D_1 and the clustering procedure, respectively. In step 6, a clustering partition is obtained from it. Finally, in steps 7 and 8, a dissimilarity matrix $\mathcal{M}$ is computed. Each entry $\mathcal{M}_{i,j}$ of $\mathcal{M}$ is defined as follows:

$$\mathcal{M}_{i,j} = 1 - \frac{M_{i,j}}{I_{i,j}}, \tag{3.1}$$

where $M_{i,j}$ is the number of times in which items i and j are in the same cluster and $I_{i,j}$ is the number of times in which items i and j are in the same learning

```
BagClust1($D_0, H_c, \beta, < C_1 >, k$)

for $i \leftarrow 0$ to $H_c$ do
  begin
  1. Generate (via a bootstrap) a data matrix $D_1$
  2. Let $P_1$ and $P_2$ be the corresponding partitions of $D_0$ and $D_1$
     into $k$ cluster with the use of $C_1$.
  3. Permute the elements assigned to the partition $P_2$ so that there is
     maximum overlap with $P_1$
  4. Let $O_c$ be the number of overlapping elements
  end
return ($O_c$)
```

Figure 3.5: The *BagClust1* procedure.

dataset. The dissimilarity matrix $\mathcal{M}$ is then used as input to a clustering procedure in order to obtain a partition. For convenience of the reader the *BagClust2* procedure is given in Fig. 3.6.

3.2.2 The Stability Measure Paradigm

In this section, the main paradigm of internal stability methods is described. It is best presented as a procedure, reported in Fig. 3.7. Its macro operations are described in abstract form in Fig. 3.8, while its basic steps are described below.
For each k in the range $[k_{min}, k_{max}]$ the paradigm collects the statistics S^k computed by the Stability_Statistic procedure, then a concise description R^k of the statistic S^k is computed via the Synopsis procedure. Finally, an explicit or implicit prediction of the value of k^* is computed by Significance_Analysis and it is given as output.

In the remaining part of this section, only two examples of the Stability Measure paradigm are detailed. The other incarnations proposed in the literature are discussed in the remaining part of the chapter.

3.2.2.1 Instances

The presentation of methods in this section is organized as follows: for each method, the input parameters setup is first described (see Fig. 3.2), then the Stability_Statistic and the Stability_Measure procedures are detailed.

- *Model Explorer* by Ben-Hur et al. [16] (`ME` for short) is the simplest incarnation of the Stability Measure paradigm and it can be derived in the following way.

```
BagClust2($D_0, H_c, \beta, < C_1 >, k$)

for $i \leftarrow 0$ to $H_c$ do
  begin
1. Generate (via a bootstrap) a data matrix $D_1$
2. Let $P_1$ be the partition of $D_1$ into $k$ clusters with the use of $C_1$
3. Compute $\mathcal{M}$ as in (3.1)
  end
return ($\mathcal{M}$)
```

Figure 3.6: The *BagClust2* procedure.

```
Stability_Measure($k_{min}, k_{max}, D, H, \alpha, \beta, < C_1, C_2, \ldots, C_t >$)

 for $k \leftarrow k_{min}$ to $k_{max}$ do
   begin
1.    $S^k \leftarrow$ Stability_Statistics($D, H, \alpha, \beta, < C_1, C_2, \ldots, C_t >, k$)
2.    $R^k \leftarrow$ Synopsis($S^k$)
   end
3. $k^* \leftarrow$ Significance_Analysis($R^{k_{min}}, \ldots, R^{k_{max}}$)
return ($k^*$)
```

Figure 3.7: The Stability_Measure procedure.

MACRO OPERATIONS

- SYNOPSIS: it takes as input a statistic and returns as output a concise description of it.
- SIGNIFICANCE_ANALYSIS: it takes as input all the statistics/information collected as returned by the SYNOPSIS procedure. It computes the significance level of each statistic. It returns as output, explicitly or implicitly, a prediction about k^*. For instance, an implicit prediction of the value of k^* can be the plot of a histogram or of a curve, as in many methods described in the next section.

Figure 3.8: List of the macro operations used in STABILITY_MEASURE procedure.

– *Input parameters setup*: $\hat{H}_c$ is used as test, for a given number of iteration c, $\alpha = 0$, $\beta \in [0.6, 0.9]$ and the set of procedures $< C_1, C_2, \ldots, C_t >$ consists only of one clustering algorithm C_1.
– *The* STATISTICS_STABILITY *procedure*: in step 1, D_1 and D_2 are generated by two DGP procedures, where each procedure is an instance of subsampling (see Section 1.5.1). Since $\alpha = 0$, the SPLIT procedure copies those datasets into the corresponding learning datasets, while steps 3 and 4 are not performed. In step 5, the graph $\hat{G}$, obtained as output of the ASSIGN procedure, encodes two relations: $< D_{L,1}, C_1 >$ and $< D_{L,2}, C_1 >$. In step 6, two clustering solutions P_1 and P_2 are obtained from $< D_{L,1}, C_1 >$ and $< D_{L,2}, C_1 >$, respectively. The COLLECT_STATISTIC procedure computes the level of agreement between the two partitions via an external index (see Section 2.1), but restricted to the common elements of D_1 and D_2. In step 8, this level of agreement is stored into a one dimensional array S^k. That is, for each iteration of the **while** loop, the value returned by the external index is stored in the corresponding entry of S^k.
– *The* STABILITY_MEASURE *procedure*: for each k, it computes the array S^k via the STATISTIC_STABILITY procedure while the SYNOPSIS procedure performs a copy of the collected statistic. Finally, the SIGNIFICANCE_ANALYSIS procedure provides an implicit estimation of k^*: each $k \in [k_{min}, k_{max}]$, R^k and its values are histogrammed separately. Then, the optimal number of clusters k^* is predicted to be the lowest value of k such that the R^k value distribution is close to one and R^{k+1} value distribution is in a wider range of values. An example of the number of clusters prediction is given in Fig. 3.10, where `ME` is computed on the dataset of Fig. 1.4(a) with K-means-R (see Section 1.2.2) for $k \in [2, 5]$.

For convenience of the reader the `ME` procedure is given in Fig. 3.9.

- *MOSRAM* by Bertoni and Valentini [21] is strongly related to `ME`, where the most significant change is in the SIGNIFICANCE_ANALYSIS procedure. Indeed, it estimates automatically the "true" number of clusters, and in addition, it detects significant and possibly multi-level structures simultaneously present in D (e.g. hierarchical structures - see Fig. 3.12). It can be derived from the Stability Measure paradigm as follows.

```
ME($H_c$, $< C_1 >$, $D$, $k_{max}$)

 for $k \leftarrow 2$ to $k_{max}$ do
   begin
     for $i \leftarrow 1$ to $H_c$ do
       begin
1.      Generate (via subsampling) two data matrices $D_1$ and $D_2$
2.      Let $P_1$ and $P_2$ be the corresponding partitions of $D_1$ and $D_2$
        into $k$ cluster with the use of $C_1$, respectively
3.      Let $S^k(i)$ be the level of agreement between $P_1$ and $P_2$ via an
        external index but restricted to the common elements of $D_1$ and $D_2$
       end
   end
 4. Plot separately the histogram of $S^k$ values and return a prediction for $k^*$
```

Figure 3.9: The `ME` procedure

- *The input parameters setup*: as in `ME`.

- *The* Statistics_Stability *procedure*: it is the same proposed in `ME`, except that the two DGP procedures performed in step 1 are both an instance of randomized mapping (see Section 1.5.3).

- *The* Stability_Measure *procedure*: in step 2, each R^k given as output by the Synopsis procedure is an average of the statistics S^k computed in step 1. Intuitively, if the value of R^k is close to 1, then the clustering solution is stable. Moreover, in order to detect significant and possibly multi-level structures that are simultaneously present in D, a statistical hypothesis test is applied. The Significance_Analysis procedure performs a χ^2-based test in order to estimate k^* as follows. Let $R = \{R^{k_{min}}, \ldots, R^{k_{max}}\}$ and let τ be a significance level. The null hypothesis H_0 considers the set of k-clusterings as equally reliable, while the alternative hypothesis H_1 considers the set of k-clusterings as not equally reliable. When H_0 is rejected at τ significance level, it means that at least one k-clustering significantly differs from the others. The procedure sorts the values in R, and a χ^2-based test is repeated until no significant difference is detected or the only remaining clustering is the top-ranked in R. At each iteration, if a significant difference is detected, the bottom-ranked value is removed from the set R. Therefore, the Significance_Analysis gives as output the set of the remaining (top sorted) k-clusterings that corresponds to the set of the estimate "true" number of clusters (at τ significance level). For convenience of the reader the *MOSRAM* procedure is given in Fig. 3.11.

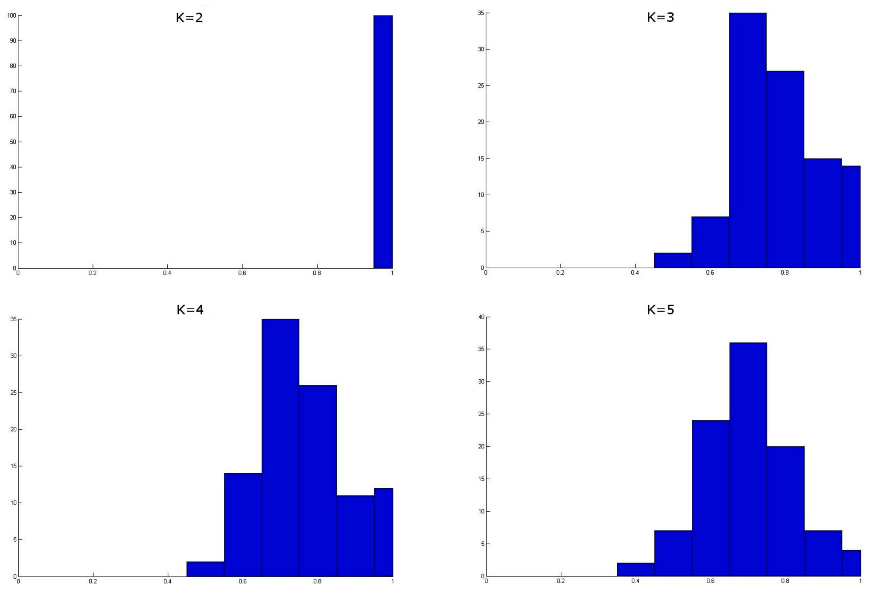

Figure 3.10: The histograms plotting the R^k values distribution for increasing values of k. The prediction of k^* correspond to correct number of cluster, i.e., $k^* = 2$.

3.3 Further Instances of the Stability Measure Paradigm

In this section several incarnations of the Stability Measure paradigm are detailed, describing for each method the input parameters and macro operations listed in Figs. 3.2, 3.3 and 3.8. The section follows the same organization of Section 3.2.2.1.

3.3.1 Consensus Clustering

Consensus Clustering by Monti et al. [122] (`Consensus` for short) is a reference method in internal validation measures, with a prediction power far better than other established methods [62, 122].

- *The input parameters setup*: it uses the same input parameters setup of `ME` for the STATISTIC_STABILITY procedure (see Fig. 3.2). Moreover, as in `ME`, the DGP procedure is an instance of subsampling (see Section 1.5.1).

- *The* STATISTICS_STABILITY *procedure*: it is strongly related to *BagClust2*, where the most significant change is that the matrix $\mathcal{M}$ is a similarity instead of a dissimilarity matrix.

- *The* STABILITY_MEASURE *procedure*: as in `ME`, the SYNOPSIS procedure performs a copy of the collected statistic and the SIGNIFICANCE_ANALYSIS procedure provides an implicit estimation of k^*, as detailed next. Based on the collected statistics, for each k, Monti et al. define a value $A(k)$ measuring the level of stability in cluster assignments, as reflected by the matrix $\mathcal{M}$. Formally,

```
MOSRAM($H_c$, $< C_1 >$, $D$, $k_{max}$)

for $k \leftarrow 2$ to $k_{max}$ do
  begin
    for $i \leftarrow 1$ to $H_c$ do
      begin
1.      Generate (via randomized mapping) two data matrices $D_1$ and $D_2$
2.      Let $P_1$ and $P_2$ be the corresponding partitions of $D_1$ and $D_2$
        into $k$ cluster with the use of $C_1$, respectively
3.      Let $S^k(i)$ be the level of agreement between $P_1$ and $P_2$ via an
        external index but restricted to the common elements of $D_1$ and $D_2$
      end
  end
4. Perform a $\chi^2$-based test in order to estimate $k^*$
```

Figure 3.11: The *MOSRAM* procedure

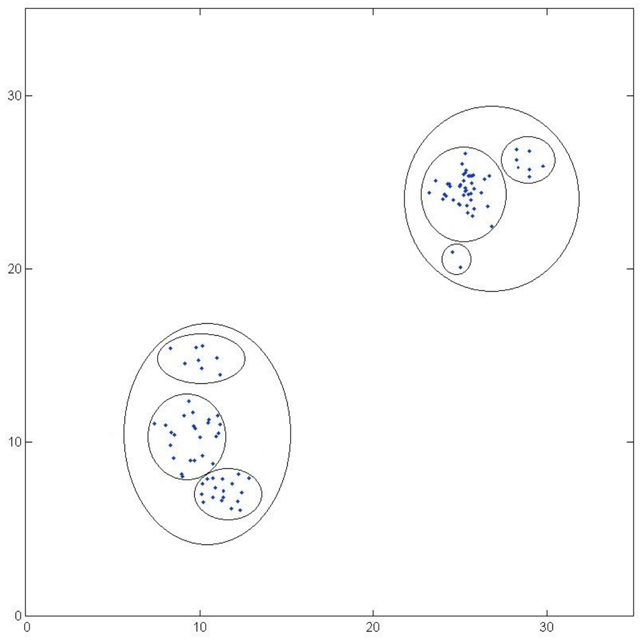

Figure 3.12: An example of hierarchical structures in a dataset.

$$A(k) = \sum_{i=2}^{n} [x_i - x_{i-1}] CDF(x_i)$$

where CDF is the empirical cumulative distribution defined over the range $[0,1]$, as follows:

$$CDF(c) = \frac{\sum_{i<j} l\{\mathcal{M}_{i,j} \leq c\}}{n(n-1)/2}$$

with l equal to 1 if the condition is true and 0 otherwise. Finally, based on $A(k)$, one can define:

$$\Delta(k) = \begin{cases} A(k) & k = 2, \\ \frac{A(k+1)-A(k)}{A(k)} & k > 2. \end{cases}$$

Moreover, Monti et al. suggest the use of the function Δ' for non-hierarchical algorithms. It is defined as Δ but one uses $A'(k) = \max_{k' \in [2,k]} A(k')$. The reason is the following: $A(k)$ is a value that is expected to behaves like a non-decreasing function of k, for hierarchical algorithms. Therefore $\Delta(k)$ would be expected to be positive or, when negative, not too far from zero. Such a monotonicity of $A(k)$ is not expected for non-hierarchical algorithms. Therefore, another definition of Δ is needed to ensure a behavior of this function analogous to the hierarchical algorithms.

Assuming that one has computed the Δ curve for a given dataset, the value of k^* can be obtained by using the following intuitive idea, also based on experimental observations.

(i) For each $k \leq k^*$, the area $A(k)$ markedly increases. This results in an analogous pronounced decrease of the Δ curve.

(ii) For $k > k^*$, the area $A(k)$ has no meaningful increases. This results in a stable plot of the Δ curve.

From this behavior, the "rule of thumb" to identify k^* is: take as k^* the abscissa corresponding to the smallest non-negative value where the curve starts to stabilize; that is, no big variation in the curve takes place from that point on. However, the behavior of the Δ curve could give an ambiguous estimation of k^*. Therefore, it is advisable to combine the information given by the Δ curve with an estimation provided by the plot of the CDF curves. In this latter estimation, k^* is the value of k where the area under the CDF curves does not change more, i.e., the gap between the curves stay almost constant. An example is given in Fig. 3.14.

The same considerations and rule applies to the prediction of k^* via Δ'. However, as the experiments performed in Chapter 6 bring to light, for the partitional algorithms used in this dissertation, Δ' displays nearly the same monotonicity properties of Δ, when used on hierarchical algorithms. The end result is that Δ can be used for both types of algorithms. It is worth pointing out that, to the best of our knowledge, Monti et al. defined the function Δ', but they did not experiment with it, since their experimentation was limited to hierarchical algorithms. For convenience of the reader the `Consensus` procedure is given in Fig. 3.13.

For completeness, it may be of interest to the reader to report that Brunet et al. [30] propose a different approach to estimate k^* based on the dispersion of the matrix $\mathcal{M}$. Indeed, Brunet et al. compute the cophenetic correlation coefficient ρ [87] for each k.

```
CONSENSUS($H_c, < C_1 >, D, k_{max}$)

 for $k \leftarrow 2$ to $k_{max}$ do
   begin
     for $i \leftarrow 1$ to $H_c$ do
       begin
1.       Generate (via a subsampling) a data matrix $D_1$
2.       Let $P_1$ be the partition of $D_1$ into $k$ clusters with the use of $C_1$
3.       Based on $P_1$ compute the connectivity matrix $M_i^k$
       end
4.   Compute the consensus matrix $\mathcal{M}^k$
   end
5. Based on the $k_{max} - 1$ consensus matrices, return a prediction for $k^*$
```

Figure 3.13: The CONSENSUS procedure

Based on the observation on how the value of ρ changes as k increases, the "rule of thumb" to identify k^* is: take as k^* the abscissa corresponding to the value where the curve starts to fall.

Finally, it can be useful to observe that the matrix $\mathcal{M}$ can be naturally thought of as a similarity measure. Accordingly, via standard techniques, it can be transformed into a (pseudo) distance matrix that can be used by clustering algorithms as in *BagClust2* in order to solve problem (**Q.2**).

3.3.2 Levine and Domany

This method is due to Levine and Domany [110].

- *The input parameters setup*: as in `ME`.

- *The* STATISTICS_STABILITY *procedure*: it is strongly related to *BagClust2*. Indeed, for each iteration, the method computes as statistic a connectivity matrix in which each entry is 1, if the two elements are in the same cluster and 0 otherwise. Moreover, the collected statistic S^k is a set of matrices, $S^k = \{S_0^k, \dots, S_c^k\}$. Matrix S_0^k corresponds to the connectivity matrix for D_0, and matrix S_i^k, for $1 \le i \le c$, corresponds to the connectivity matrix for the dataset D_1 generated by the DGP procedures at the corresponding iteration.

- *The* STABILITY_MEASURE *procedure*: the SYNOPSIS procedure compares the collected statistic, via the following formula:

$$R^k = \ll \delta_{S_0^k, S_i^k} \gg \tag{3.2}$$

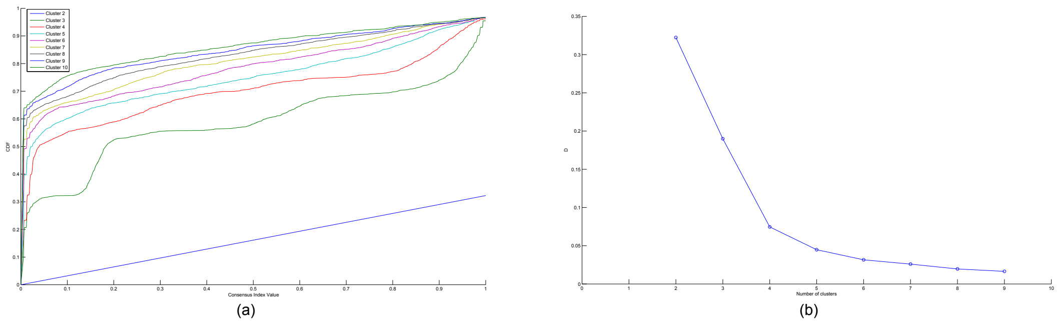

Figure 3.14: An example of number of cluster prediction with the use of `Consensus`. The experiment is derived from the dataset of Fig. 1.4(a) with $k^* = 2$, with use of the K-means-R clustering algorithm. The plots of the CDF curves is shown in (a), yielding a monotonically increasing value of A, as a function of k. The plot of the Δ curve is shown in (b), where the flattening effect corresponding to $k = 4$, while the gap of the area under the CDF curves is evident for $k \geq k^* = 2$.

where $\ll \cdot \gg$ is a twofold averaging. That is, for each S_i^k, an average is computed over all pairs which are in the same cluster in the original dataset and have been both selected in the same resample. Then, an average for all S_i^k is computed. The SIGNIFICANCE_ANALYSIS procedure gives as output k^* as the value of k with the local maximum of R^k, for $k \in [k_{min}, k_{max}]$.

For convenience of the reader the procedure proposed by Levine and Domany is given in Fig. 3.15.

3.3.3 Clest

`Clest`, proposed by Dudoit and Fridlyand [48], generalizes in many aspects *replicating analysis* by Breckenridge (see Section 3.2.1). It can be regarded as a clever combination of hypothesis testing and resampling techniques. It estimates k^* by iterating the following: randomly partition the original dataset in a *learning* set and *training* set. The learning set is used to build a classifier $\mathcal{C}$ for the data, then to be used to derive "gold standard" partitions of the training set. That is, the classifier is assumed to be a reliable model for the data. It is then used to assess the quality of the partitions of the training set obtained by a given clustering algorithm.

- *The input parameters setup*: it uses the same input parameters of *replicating analysis*, except for the test condition H, where in this case c iterations of the **while** loop are allowed, for a given integer $c > 1$.

- *The* STATISTICS_STABILITY *procedure*: it corresponds to the *replicating analysis*. Therefore, the set S^k of records is a one dimensional array, in which each entry stores the value of the external index for the corresponding iteration.

- *The* STABILITY_MEASURE *procedure*: the SYNOPSIS procedure computes R^k as the median of the values stored in S^k. The SIGNIFICANCE_ANALYSIS procedure proposed in `Clest` is best presented as a procedure which is given in Fig. 3.16 and it is outlined next. The first step of the procedure generates a new dataset via the

```
LD01($H, < C_1 >, D, k_{max}$)

 for $k \leftarrow 2$ to $k_{max}$ do
   begin
 1.  Let $P_1$ be the partition of $D$ into $k$ clusters with the use of $C_1$
 2.  Based on $P_1$ compute the connectivity matrix $S_0^k$
       for $i \leftarrow 1$ to $H$ do
         begin
 3.        Generate (via a subsampling) a data matrix $D_1$
 4.        Let $P_1$ be the partition of $D_1$ into $k$ clusters with the use of $C_1$
 5.        Based on $P_1$ compute the connectivity matrix $S_i^k$
         end
 6.  Compute $R^k$, as defined in (3.2)
   end
```

Figure 3.15: The Levine and Domany procedure

DGP procedure that in this case it is an instance of null models (see Section 1.4), the p_{max} is a "significance level" threshold and d_{min} is a minimum allowed difference between "computed and expected" values. It is worth pointing out that the SIGNIFICANCE_ANALYSIS procedure provides an explicit prediction of k^*.

For convenience of the reader the `Clest` procedure is given in Fig. 3.17. It is worth pointing out that steps 1-7 correspond to the STABILITY_STATISTICS detailed above. Step 8 is the SYNOPSIS call in the STABILITY_MEASURE procedure, and the next steps corresponds to the SIGNIFICANCE_ANALYSIS procedure described above (see Fig. 3.16).

3.3.4 Roth et al.

In analogy with `Clest`, this method, by Roth et al. [147], also generalizes *replicating analysis*.

- *The input parameters setup*: as in `Clest`.

- *The* STATISTICS_STABILITY *procedure*: steps 1-6 that are the same as in *replicating analysis*. Recall from that latter procedure, that P_1 and P_2 are two partitions obtained by a classifier and a clustering algorithm, respectively. The COLLECT_STATISTIC procedure takes as input P_1 and P_2 and generates a new partition by computing a minimum weighted perfect bipartite matching [129]. Therefore, assuming P_1 as a correct solution, COLLECT_STATISTIC gives as output the number of misclassified elements, normalized with respect to the case in which the prediction is random.

```
SIGNIFICANCE_ANALYSIS($R^{k_{min}}, \ldots, R^{k_{max}}$)

for $i \leftarrow 0$ to $B_0$ do
  begin
1.  $D^i \leftarrow \mathrm{DGP}(D)$
2.  for $k \leftarrow k_{min}$ to $k_{max}$ do
      begin
3.      $S^k \leftarrow$ STABILITY_STATISTICS($D^i, H, \beta, < C_1, C_2, \ldots, C_t >, k$)
4.      $R_i^k \leftarrow$ SYNOPSIS($S^k$)
      end
  end
5. for $k \leftarrow k_{min}$ to $k_{max}$ do
    begin
6.    $t_k^0 \leftarrow$ Compute the average of the $R_i^k$ values
7.    $p_k \leftarrow$ Compute the p-value of $R^k$
8.    $d_k \leftarrow R^k - t_k 0$
    end
9. Define a set $K = \{k_{min} \leq k \leq k_{max} : p_k \leq p_{max}$ and $d_k \geq d_{min}\}$
10. if $K = \emptyset$
  then $k^* \leftarrow 1$
  else $k^* \leftarrow \arg\max_{k \in K} d_k$
return $(k^*)$
```

Figure 3.16: Implementation of the SIGNIFICANCE_ANALYSIS procedure proposed by Dudoit and Fridlyand for `Clest`.

```
CLEST($B_0, H_c, < C_1, C_2 >, E, D, k_{max}, p_{max}, d_{min}$)

for $k \leftarrow 2$ to $k_{max}$ do
  begin
1.  for $h \leftarrow 1$ to $H_c$ do
      begin
2.      Split the input dataset in $D_L$ and $D_T$, the learning and training sets,
         respectively
3.      Train the classifier $C_1$on $D_T$
6.      Partition $D_L$ into $k$ clusters by $C_1$ and $C_2$ in order to obtain
         $P_1$ and $P_2$, respectively.
7.      $m_{k,h} = E(P_1, P_2)$
      end
8.   $t_k \leftarrow median(m_{k,1}, \ldots, m_{k,H})$
     for $b \leftarrow 1$ to $B_0$ do
       begin
9.      Generate (via a null model), a data matrix $D^b$
10.      Repeat steps 1-8 on $D^b$
       end
11.  Compute the average of these H statistics, and denote it with $t_k^0$
12.  Compute the p-value $p_k$ of $t_k$
13.  $d_k \leftarrow t_k - t_k^0$
  end
14. $K \leftarrow \{2 \leq k \leq k_{max} : p_k \leq p_{max}$ and $d_k \geq d_{min}\}$
if $K = \emptyset$
  then $k^* \leftarrow 1$
  else $k^* \leftarrow \arg\max_{k \in K} d_k$
return $(k^*)$
```

Figure 3.17: The CLEST procedure.

```
RLBB02($H_c$, $< C_1, C_2 >$, $D$, $k_{max}$)

 for $k \leftarrow k_{min}$ to $k_{max}$ do
   begin
    for $i \leftarrow 0$ to $H_c$ do
        begin
1.         Split the input dataset in $D_L$ and $D_T$, the learning and training sets,
            respectively.
3.         Train the classifier $C_1$on $D_T$
5.         Partition $D_L$ into $k$ clusters by $C_1$ and $C_2$ in order to obtain
           $P_1$ and $P_2$, respectively.
6.         Find the correct permutation by the minimum
           weighted perfect bipartite matching
7.         Normalize w.r.t. the random
8.         Compute the expected (in)-stability value
        end
   end
```

Figure 3.18: The Roth et al. procedure.

- *The* Stability_Measure *procedure*: the Synopsis procedure computes the average over the assignment cost and it computes the "expected (in)-stability" value defined as the expectation with respect to the two different datasets. Finally, the Significance_Analysis procedure gives as output the value of k with the minimum "expected (in)-stability" value as k^*.

For convenience of the reader, the procedure proposed by Roth et al. is given in Fig. 3.18.

3.3.5 A Special Case: the Gap Statistics

Although `Gap` (see Section 2.2.2) is not an internal stability measure, the GP procedure (see Fig. 2.4) can be derived from the Stability Measure paradigm as follows.

- *The input parameters setup*: $\hat{H}_1$, $\alpha = 0$, β is not relevant and the set of clustering procedures $< C_1, C_2, \ldots, C_t >$ consists of only one clustering algorithm C_1.

- *The* Statistics_Stability *procedure*: only steps 2 and 5-8 are performed. The Collect_Statistic procedure takes as input a partition P_1 relative to the dataset D and it gives as output the `WCSS` value (see Section 2.2.1.1).

- *The* Stability_Measure *procedure*: the Synopsis procedures return a copy of the `WCSS` values. Finally, the Significance_Analysis procedure is best presented as a

```
SIGNIFICANCE_ANALYSIS($R^{k_{min}}, \dots, R^{k_{max}}$)

 for $k \leftarrow k_{min}$ to $k_{max}$ do
   begin
     for $i \leftarrow 0$ to $B_0$ do
         begin
           $D^i \leftarrow \mathrm{DGP}(D)$
           $R_i^k \leftarrow$ STABILITY_STATISTICS($D^i, H, \beta, < C_1 >, k$)
         end
      Compute $Gap(k) = \frac{1}{B_0} \sum_{i=1}^{B_0} R_i^k - R^k$.
      Compute the standard deviation $sd(k)$of the set of numbers $\{R_1^k, \dots, R_{B_0}^k\}$
      $s(k) = \left(\sqrt{1 + \frac{1}{B_0}}\right) sd(k)$.
   end
 $k^*$ is the first value of $k$ such that $Gap(k) \geq Gap(k+1) - s(k+1)$.
 return ($k^*$)
```

Figure 3.19: Implementation of the SIGNIFICANCE_ANALYSIS procedure for `Gap`.

procedure which is given in Fig. 3.19, where in the first step the procedure generates a new dataset via the DGP procedure that in this case it is an instance of null models (see Section 1.4). Intuitively, in the remaining steps, it compares the value of `WCSS` obtained for D and for the null model datasets. The prediction of k^* is based on running a certain number of times the procedure STABILITY_MEASURE taking then the most frequent outcome as the prediction.

Chapter 4

Non-negative Matrix Factorization

This chapter describes one of the methodologies that has gained prominence in the data analysis literature: Non-negative Matrix Factorization (NMF for short). In particular, the mathematical formulation of NMF and some algorithms that have been developed to compute it are presented. Moreover, some applications of NMF are also discussed. It is worth pointing out that, for this dissertation, NMF is of interest as a clustering algorithm. In fact, in the next chapter, its first benchmarking on microarray data is presented.

4.1 Overview

One common ground on which many data analysis methods rest is to replace the original data by a lower dimensional representation obtained via subspace approximations [32, 79, 144, 158, 174]. The goal is to explain the observed data using a limited number of basis components, which when combined together, approximate the original data as accurately as possible. This concise description of the data allows to find possible structure in them. Indeed, a meaningful dimensionality reduction can be achieved only if the data has common underlying regularities and patterns. Matrix factorization and principal component analysis are two of the many classical methods used to accomplish both the goals of reducing the number of variables and detecting structure underlying the data. While some of those techniques have been reviewed in Section 1.5.3, NMF is singled-out in this chapter since it is a substantial contribution to this area: the pioneering paper by Lin and Seung has been followed-up by extensions and uses of NMF in a broad range of domains. In particular, molecular biology applications, as described in the survey of Devarajan [41], image processing [70, 107] and text mining [107, 123].

This chapter is organized as follows. Section 4.2 provides a formalization of matrix factorization. Sections 4.3 and 4.4 provide the general scheme and different algorithms for NMF, respectively. Finally, in Section 4.4, several applications of NMF to image processing, text mining and molecular biology are briefly described.

4.2 Matrix Factorization: A Basic Mathematical Formulation

In this section, a general statement of Matrix Factorization (MF for short) is given, then two restricted versions of it are presented: Positive Matrix Factorization (PMF for short) and the already mentioned NMF.

MF. Let V be matrix of size $m \times n$. Usually, when V is a data matrix, m denotes the number of features and n denotes the number of items in Σ. Given V and an integer $r < \min\{n, m\}$, one wants to find two matrix factors W and H of size $m \times r$ and $r \times n$, respectively, such that:

$$V \approx W \times H. \tag{4.1}$$

One has that $v \approx Wh$, where v and h are homologous columns in V and H, respectively. That is, each column v is approximated by a linear combination of the columns of W, weighted by the components of h. Therefore, W can be regarded as containing a basis.

PMF. It is a variant of MF, by Paatero and Tapper [128], since V is constrained to be a positive matrix. One possible solution can be obtained by computing a positive low-rank approximation of $W \times H$, via an optimization of the function:

$$\min_{W,H \geq 0} \|A \circ (V - W \times H)\|_F,$$

where A is a weighted matrix whose elements are associated to the elements of V, $\circ$ is the Hadamard product and $\|\|_F$ denotes the Frobenious norm [64]. Paatero and Tapper also proposed an alternative least squares algorithm in which one of the two matrices is fixed and the optimization is solved with respect to the other one and viceversa. Later, Paatero developed a series of algorithms [125, 126, 127] using a longer product of matrices to replace the approximate $W \times H$.

NMF. A variant of PMF, NMF allows V to be non-negative, this latter being a constraint much more suitable for data analysis tasks. It also offers several advantages [58, 107]:

1. The constraint that the matrix factors are non-negative allows for their intuitive interpretation as real underlying components within the context defined by the original data. The basis components can be directly interpreted as parts or basis samples, present in different proportions in each observed sample.

2. NMF generally produces sparse results, implying that the basis and/or the mixture coefficients have only a few non-zero entries.

3. Unlike other decomposition methods such as singular value decomposition [165] or independent component analysis [14, 34], the aim of NMF is not to find components that are orthogonal or independent, therefore allowing overlaps among the components.

4.3 The General NMF Scheme

In order to compute an NMF, all the methods proposed in the literature use the same simple scheme, reported in Fig. 4.1, where the following "ingredients" have to be specified:

(i) an initialization for matrices W and H;

(ii) an update rule;

(iii) a stopping criterion.

```
NMF($V$)
  Initialize $W$ and $H$
 while $H$ do
   begin
      update $W$ and $H$
   end
```

Figure 4.1: The basic NMF procedure.

The scheme is iterative: it starts from the two initial matrices W and H, which are repeatedly updated via a fixed rule until the stopping criterion is satisfies.

For point (i), the matrices W and H are initialized at random. In that case, different runs of NMF, with the same input, are likely to produce different results. However, it is worth pointing out that sophisticated deterministic initialization methods have been proposed to choose appropriate initial values referred to as "seed NMF algorithms" [7, 28, 173]. When one uses the same initial seed, the procedure is deterministic, i.e., it always produces the same output on a given input.

With respect to points (ii) and (iii), quite many numerical algorithms have been developed for NMF [106, 108, 111, 112, 137].

For (ii), the most popular follow at least one of the following principles and techniques: alternating direction iterations, projected Newton, reduced quadratic approximation, and descent search. Correspondingly, specific implementations can be categorized into alternating least squares algorithms (ALS for short) [128], multiplicative update algorithms [107, 108] combined with gradient descent search, and hybrid algorithms [19, 137]. For the interested reader, it is worth pointing out that a general assessments of these methods can be found in [103, 166]. In the next section, a brief description of the gradient descent and ALS methods is provided.

As for (iii), the most "popular" stopping criteria are: a fixed number of iterations, of a suitably defined matrix, referred to as consensus (see [30] for formal definitions) and stationarity of the objective function value.

4.4 The NMF Procedure and Its Variants

As anticipated, two of the most popular strategies for the NMF computation are detailed here. Moreover, other approaches specific to particular applications are briefly described. Finally, a list of the software available for the computation of NMF is reported.

4.4.1 Multiplicative Update Rules and Gradient Descent

Gradient descent is an optimization strategy widely used in the literature [157, 162, 177]. Let F be a function to be minimized. Unfortunately, it is possible that the computation of the absolute minimum of F may be practically unfeasible. In that case, a local minimum is an "acceptable solution". With reference to Fig. 4.2, the gradient descent approach tries to compute such a local minimum as follows: the algorithm starts from a random point and moves to a successive point by minimizing along the local downhill gradient between the two points. This process is iterated until a minimum of the objective function is reached.

Lee and Seung propose the following optimization function :

$$f(W,H) = \frac{1}{2}\|V - W \times H\|^2 \tag{4.2}$$

as a convergence criterion.

In order to optimize that function, Lee and Seung propose "multiplicative update rules" in conjunction with gradient descent. As discussed in [108], the use of a multiplicative update strategy as opposed to an additive one is that the latter does not guarantee a systematic decrease in the cost function.

The algorithm starts with a random initialization of the two matrices W and H and then, at each iteration, updates them as follows:

$$H_{a,\mu} \leftarrow H_{a,\mu} \times \frac{(W^t \times V)_{a,\mu}}{(W^t \times W \times H)_{a,\mu}}$$

and

$$W_{i,a} \leftarrow W_{i,a} \times \frac{(V \times H^t)_{i,a}}{(W \times H \times H^t)_{i,a}}$$

where $1 \leq a \leq n$, $1 \leq i, \mu \leq r$ and A^t denotes the transpose of a matrix A.

Letting $\nabla_H f(W,H) = W^t \times (W \times H - V)$ and $\nabla_W f(W,H) = (W \times H - V) \times H^t$, the above update rules can be rewritten as follows:

$$H_{a,\mu} \leftarrow H_{a,\mu} - \frac{H_{a,\mu}}{(W^t \times W \times H)_{a,\mu}} \times \nabla_H f(W,H) \tag{4.3}$$

and

$$W_{i,a} \leftarrow W_{i,a} - \frac{W_{i,a}}{(W \times H \times H^t)_{i,a}} \times \nabla_W f(W,H) \tag{4.4}$$

It is worth pointing out that the optimization function (4.2) in the general NMF model can be modified in several ways, depending on the application at hand. For instance, some penalty terms can be added in order to gain more localization or enforce sparsity, and more constraints such as sparseness can be imposed.

Two drawbacks of the multiplicative algorithms are that: (1) the denominator of the step size may be zero and (2) once an element in W or H becomes zero, it remains zero and this fact should have some effects on the correct convergence of the algorithm.

Although Lee and Seung [108] claimed the convergence of the above algorithm, Gonzalez and Zhang [66] and Lin [111, 112] independently presented numerical counter-examples, where the Lee and Seung algorithm fails to approach a stationary point, i.e., to converge to a local minimum.

Moreover, Lin [112] proposed to modify the Lee and Seung method, in order to solve the convergence problem, by keeping the same objective function, while update rules (4.3) and (4.4) become:

$$H_{a,\mu} \leftarrow H_{a,\mu} - \frac{\overline{H}_{a,\mu}}{(W^t \times W \times \overline{H})_{a,\mu} + \delta} \times \nabla_H f(W,H)$$

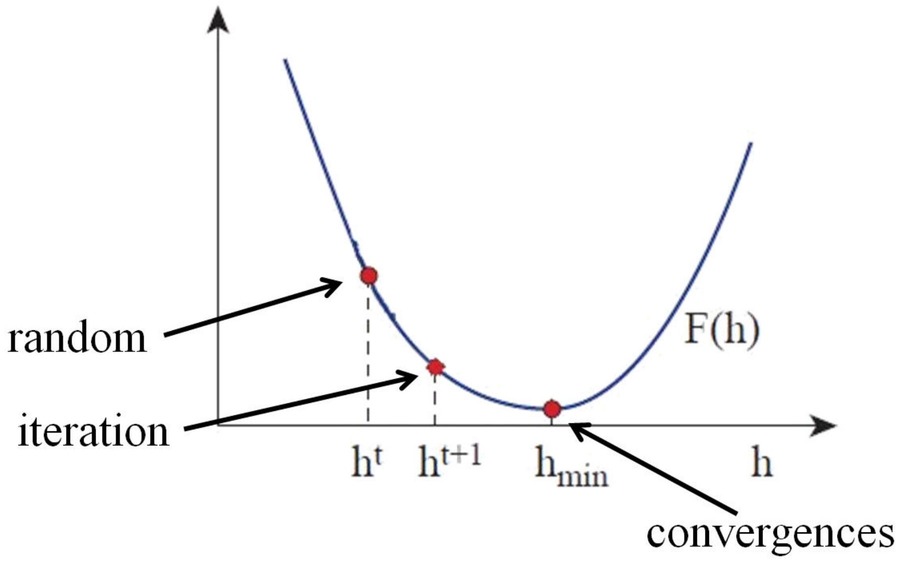

Figure 4.2: An example of gradient descend. Part of this figure is taken from [108].

and

$$W_{i,a} \leftarrow W_{i,a} - \frac{\overline{W}_{i,a}}{(\overline{W} \times H \times H^t)_{i,a} + \delta} \times \nabla_W f(W, H)$$

Both ε and δ are pre-defined small positive constants,

$$\overline{H}_{a,\mu} = \begin{cases} H_{a,\mu} & if\, \nabla_H f(W,H)_{a,\mu} \geq 0, \\ \max(H_{a,\mu}, \varepsilon) & otherwise, \end{cases}$$

and $\overline{W}_{i,a}$ is defined in analogy with $\overline{H}_{a,\mu}$.

Moreover, Lin also proposed to normalize W so that the sum of its columns is one. With respect to the Lee and Seung method, this modified version requires same extra operations (e.g. to compute $\overline{H}$ and $\overline{W}$), but that has no substantial effect on the complexity of the procedure.

Lin [112] proves the convergence of this modified algorithm, via the following theorem.

Theorem 2. *The update sequence of W and H has at least one limit point.*

Proof. Let W^k and H^k be the matrices at iteration k. It suffices to prove that (W^k, H^k), $k = 1, \ldots, \infty$, are in a closed and bounded set. Since W^k is normalized, one needs to show that H^k is bounded. Otherwise, there is a component $H_{a,\mu}$ and an infinite index set κ such that

$$\lim_{k\in\kappa} H^k_{a,\mu} \to \infty, \quad H^k_{a,\mu} < H^{k+1}_{a,\mu}, \quad \forall k \in \kappa, \tag{4.5}$$

and exist $\forall i$

$$\lim_{k\in\kappa, k\to\infty} W^k_{i,a}.$$

One must have that $W_{i,a} = 0$, $\forall i$. Otherwise, there is an index i such that

$$\lim_{k\in\kappa,k\to\infty} (W^k \times H^k)_{i,j} \geq \lim_{k\in\kappa,k\to\infty} W^k_{i,a} H^k_{a,j} = \infty.$$

Then,

$$\lim_{k\to\infty} f(W^k, H^k) \geq \lim_{k\in\kappa, k\to\infty} \left\| V_{i,j} - (W^k \times H^k)_{i,j} \right\|^2 = \infty$$

contradicting that $f(W^k, H^k)$ is strictly decreasing. Since the sum of the columns of W^k is either one or zero, $W_{i,a} = 0$, $\forall i$ implies

$$W_{i,a} = 0, \quad \forall i \quad \forall k \in \kappa \text{ large enough.} \tag{4.6}$$

Then,

$$\nabla_H f(W^k, H^k)_{a,\mu} = 0, \quad \forall k \in \kappa, \text{ so } H^{k,n}_{a,\mu} = H^k_{a,\mu}, \quad \forall k \in \kappa. \tag{4.7}$$

Moreover, by (4.6), one can prove that

$$\lim_{k\in\kappa, k\to\infty} W^{k,n}_{i,a} = \lim_{k\in\kappa, k\to\infty} W^k_{i,a} = 0, \forall i$$

Thus, in normalizing $(W^{k,n}, H^{k,n})$, $H^{k,n}$'s a-th row is either unchanged or decreased. With (4.7),

$$H^{k+1}_{a,\mu} \leq H^k_{a,\mu}, \forall k \in \kappa$$

an inequality contradicting (4.5). Thus, H^k is bounded. Therefore, $\{W^k, H^k\}$ is in a compact set, so there is at least one convergent sub-sequence. □

4.4.2 Alternating Least Squares Algorithms

In this class of algorithms, a least squares step is followed by another least squares step in an alternating way. That is, the algorithm updates in alternation W and H by solving a distinct matrix equations for each, as detailed in procedure ALS reported in Fig. 4.3. In that procedure, in order to keep non-negativity, a simple projection step is performed to set all negative elements resulting from the least squares computation to zero. Moreover, some additional flexibility, not available in other algorithms-especially those of the multiplicative update class- is also available. For instance, as detailed above, in the multiplicative algorithms, if an element in W or H becomes 0, it must remain 0. This is restrictive, since once the algorithm starts heading down a path towards a fixed point, even if it is a poor fixed point, it must continue in that vein. The ALS algorithms is more flexible, allowing the iterative process to escape from a poor path. It is worth pointing out that some improvements to the basic ALS algorithm scheme appear in [19, 127] and that, depending on the implementation, ALS algorithms can be very fast.

4.4.3 NMF Algorithms with Application-Dependent Auxiliary Constraints

A great deal of work has been devoted to the analysis, extension, and application of NMF algorithms in science, engineering and medicine. The NMF has been cast into alternate formulations by various authors. In this section a brief survey on the state of the art is provided. One of the main improvements is to develop the algorithms by using different objective functions. Lee and Seung [108] provided an information theoretic formulation based on the Kullback-Leibler divergence [102] of V from $W \times H$. Dhillon and Sra [44] generalized the NMF methods with Bregman divergence. Cichocki et al. [36] have proposed cost functions based on Csiszár's φ-divergence. Wang et al. [171] propose a formulation

```
ALS($V$)

while convergence is not reached do
  begin
    1. Initialize $W$ at random
    2. Solve for $H$ in matrix equation $W^t \times W \times H = W^t \times V$
       and set all negative elements in $H$ to zero
    3. Solve for $W$ in matrix equation $H \times H^t \times W^t = H \times V^t$
       and set all negative elements in $W$ to zero
  end
```

Figure 4.3: The basic ALS procedure

that enforces constraints based on Fisher linear discriminant analysis for improved determination of spatially localized features. Guillamet et al. [69] have suggested the use of a diagonal weight matrix Q in a new factorization model, $V \times Q \approx W \times H \times Q$, in an attempt to compensate for feature redundancy in the columns of W. Other approaches propose alternative cost function formulations. Smoothness constraints have been used to regularize the computation of spectral features in remote sensing data [133, 137]. Chen and Cichocki [33] used temporal smoothness and spatial correlation constraints to improve the analysis of EEG data for early detection of Alzheimer's disease. Hoyer [82, 83] employed sparsity constraints on either W or H to improve local rather than global representation of data. The extension of NMF to include such auxiliary constraints is problem dependent and often reflects the need to compensate for the presence of noise or other data degradations in V.

4.4.4 Software Available

Several algorithms performing NMF have been implemented and published. The interested reader will find a compendium of them in [19]. In this dissertation, some of them are mentioned next.

Hoyer [82] provided a package that implements five different algorithms. Cichocki and Zdunek [35] produced an appealing NMFLAB package that implements a wide range of NMF algorithms, which can be combined, tested and compared via a graphical interface. However, availability only in MATLAB, a proprietary software, limits access to these packages within the scientific community. Some C/C++ implementations are also available [169], including a parallel implementation using the MPI. Recently, Gaujoux and Seoighe [58] propose a completely open-source package for the R/BioConductor platform [59], which is a well established and extensively used standard in statistical and bioinformatics research.

4.5 Some Applications

In this section, the spectrum of possible applications of NMF is highlighted. In particular, the ones in image analysis and text mining, originally used to Lee and Seung to show the validity of NMF, are detailed. Moreover, a short summary of the more recent applications of NMF in bioinformatics is also provided.

4.5.1 Image Analysis

In this domain, the use of NMF is a natural choice, since an image can be represent as a non-negative matrix. Moreover, when it is desirable to process datasets of images represented by column vectors, as composite objects or as separated parts, it is suggested that an NMF would enable the identification and classification of intrinsic "parts" that make up the object being imaged by multiple observations [46, 107]. An example is provided in Fig. 4.4, which is taken from [107]. Other work on face and image processing applications of NMF includes [69, 70, 71, 72, 104, 105, 170].

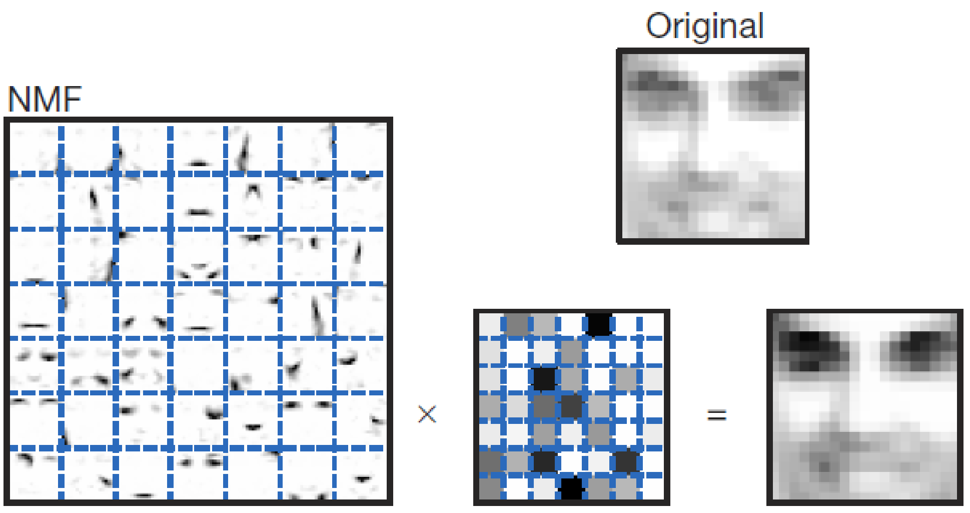

Figure 4.4: The NMF is applied to a database of 2.429 facial images, each consisting of 19×19 pixels, and constituting a matrix V. The NMF learns to represent a face as a linear combination of basis images. The NMF basis and encodings contain a large fraction of vanishing coefficients, so both the basis images and image encodings are sparse. The basis images are sparse because they are non-global and contain several versions of mouths, noses and other facial parts, where the various versions are in different locations or forms.

4.5.2 Text Mining

The use of NMF on text documents has highlighted its ability to tackle semantic issues such as synonymy or even to cluster data. An example is given in Fig. 4.5, where NMF is used to discover semantic features of 30.991 articles from the Grolier encyclopedia. For each word in a vocabulary of size 15.276, the number of occurrences is counted in each article and it is used to form the matrix V. Each column of V contains the word counts for a particular article, whereas each row of V contains the counts of a particular word in different articles. Upper left, four of the $r = 200$ semantic features (columns of W). As they are very high-dimensional vectors, each semantic feature is represented by a list of the eight words with highest frequency in that feature. The darkness of the text indicates the relative frequency of each word within a feature. With reference to Fig. 4.5 the bottom of the figure exhibits the two semantic features containing "lead" with high frequencies. Judging from the other words in the features, two different meanings of "lead" are differentiated by NMF. Right, the eight most frequent words and their counts in the encyclopedia entry on the "Constitution of the United States". This word count vector was approximated by a superposition that gave high weight to the upper two semantic features, and none to the lower two, as shown by the four shaded squares in the middle indicating the activities of H.

Other work on text mining applications of NMF includes [13, 18, 43, 134, 151, 176].

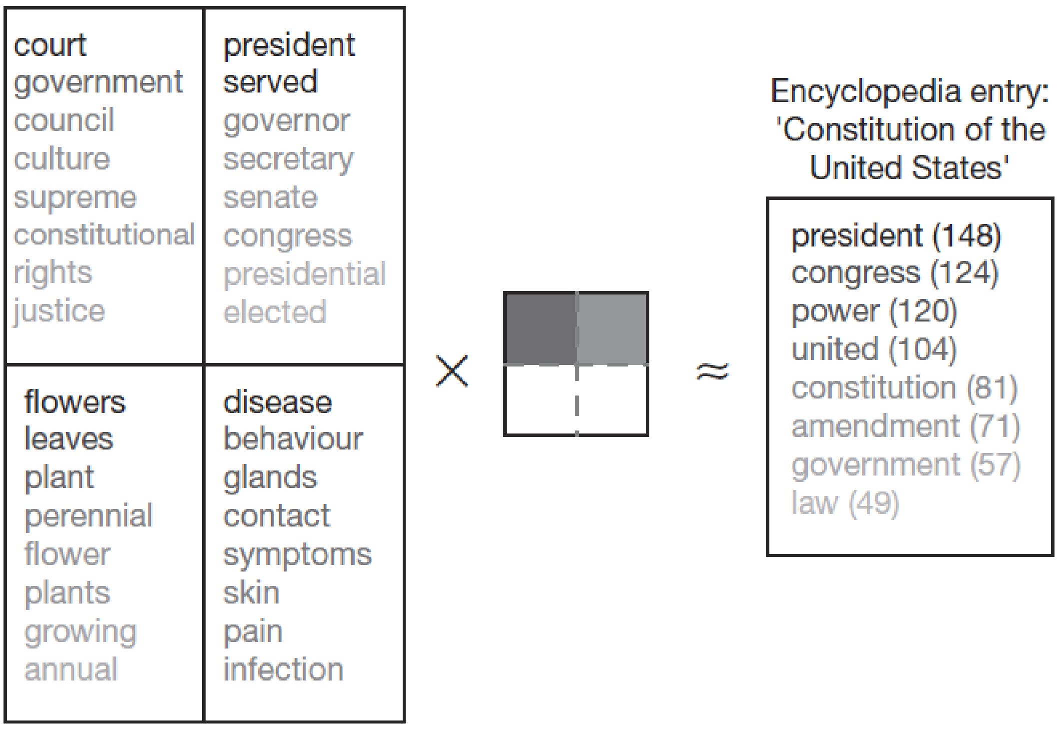

Figure 4.5: Example of NMF on text. This figure is takes from [107].

4.5.3 Bioinformatics

NMF is a very versatile pattern discovery technique that has received quite a bit of attention in the computational biology literature, as discussed in the review by Devarajan [41]. In this section, some of them are detailed.

Class Comparison and Prediction. Recently, the NMF is used in this domain. For instance, Fogel et al. [53] apply NMF to identify ordered sets of genes and utilize them in sequential analysis of variance procedures in order to identify differentially expressed genes. Okun and Priisalu [124] apply NMF, successfully, as a dimension reduction tool in conjunction with several classification methods for protein fold recognition. They report superior performance, in terms of misclassification error rate, of three classifiers based on nearest neighbor methods when applied to NMF reduced data relative to the original data. Similar approaches have been proposed in [91] and [94] for fold recognition and magnetic resonance spectroscopic imaging, respectively.

Cross-Platform and Cross-Species Characterization. The rapid advances in high-throughput technologies have resulted in the generation of independent large-scale biological datasets using different technologies in different laboratories. In this scenario, it is important to assess and interpret potential differences and similarities in these datasets in order to enable cross-platform and cross-species analysis and the possible characterization of the data. Tamayo et al. [163] describe an approach referred to as metagene projection in order to reduce noise and technological variation, while capturing invariant biological features in the data. Furthermore, this approach allows the use of prior knowledge based on existing datasets in analyzing and characterizing new data [86]. In metagene projection, the dimensionality of a given dataset is reduced using NMF, based on a pre-specified rank k factorization.

Molecular Pattern Discovery. One of the most common applications of NMF in bioninformatics, and of great interest for this thesis, is in the area of molecular pattern discovery. In particular, for gene and protein expression microarray studies, where there is lack of a priori knowledge of the expected expression patterns, for a given set of genes or phenotypes. In this area, NMF is successfully applied in order to discover biologically meaningful classes, i.e., clusters.

For instance, Kim and Tidor [98] apply NMF as a tool to cluster genes and predict functional cellular relationships in Yeast, while Heger and Holm [80] use it for the recognition of sequence patterns among related proteins. Brunet et al. [30] apply it to cancer microarray data for the elucidation of tumor subtypes. Moreover, they developed a model selection for NMF based on the consensus matrix (see Section 3.3.1) that enables the choice of the appropriate number of clusters in a dataset. Following the same notation as in Brunet et al. [30] and Devarajan [41], let V represent the outcome of a microarray experiment, where there are m samples, each composed of measurements of n genes. In this case, W and H assume two very intuitive roles. W is a matrix whose columns are "metagenes" and H is a matrix whose rows are "meta expression patterns". If one is interested in clustering the samples in r groups, as we do here, then one can place sample i in cluster j if the expression level of sample i is maximum in metagene j. That is, $H_{i,j}$ is maximum in the i-th column of H. An example is given in Figure 4.6.

Other approaches of molecular pattern discovery that use NMF [31, 38, 56, 97, 130, 131, 132], or its sparse version, have been proposed in the literature.

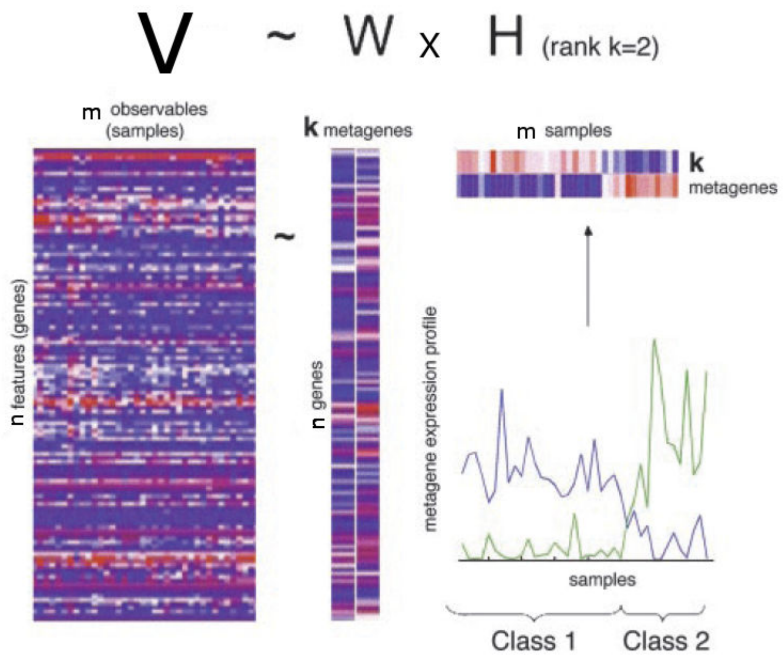

Figure 4.6: A rank-2 reduction of a DNA microarray of n genes and m samples is obtained by NMF, $V \approx W \times H$. For better visibility, H and W are shown with exaggerated width compared with original data in V, and a white line separates the two columns of W. Metagene expression levels (rows of H) are color coded by using a heat color map, from dark blue (minimum) to dark red (maximum). The same data are shown as continuous profiles below. The relative amplitudes of the two metagenes determine two classes of samples, class 1 and class 2. Here, samples have been ordered to better expose the class distinction. This figure is taken from [30].

Chapter 5

Experimental Setup and Benchmarking of NMF as a Clustering Algorithm

In this chapter, the experimental framework used in this dissertation is detailed, i.e., datasets, algorithms and hardware. Moreover, a benchmarking of NMF as a clustering algorithms on microarray data is proposed. To the best of our knowledge, it is the first one that takes into account both its ability to identify cluster structure in a dataset and the computational resources it needs for that task. A comparative analysis with some classic algorithms is also provided.

5.1 Datasets

It is useful to recall the definition of "gold solution" that naturally yields a partition of the datasets in two main categories. Technically speaking, a gold solution for a dataset is a partition of the data in a number of classes known *a priori*. Membership in a class is established by assigning the appropriate class label to each element. In less formal terms, the partition of the dataset in classes is based on external knowledge that leaves no ambiguity on the actual number of classes and on the membership of elements to classes. Although there exist real microarray datasets for which such an *a priori* division is known, in a few previous studies of relevance here, a more relaxed criterion has been adopted to allow also datasets with high quality partitions that have been inferred by analyzing the data, i.e., by the use of internal knowledge via data analysis tools such as clustering algorithms. In strict technical terms, there is a difference between the two types of "gold solutions". For their datasets, Dudoit and Fridlyand [48] elegantly make clear that difference and a closely related approach is used here.

Each dataset is a matrix, in which each row corresponds to an element to be clustered and each column to an experimental condition. In this dissertation, both microarray and simulated datasets are used. In what follows, a brief description of them is given.

5.1.1 Gene-Expression Microarray Data

The nine datasets from gene-expression microarray, together with the acronyms used in this dissertation, are reported next. For conciseness, only some relevant facts about them are mentioned. The interested reader can find additional information in Handl et al. [73], for the Leukemia dataset, in Dudoit and Fridlyand [48] for the Lymphoma and NCI60 datasets, in Monti et al. [122] for the Normal, Novartis and St.Jude datasets, finally in Di Gesú et al. [45], for the remaining ones. In all of the referenced papers, the datasets were used for validation studies. Moreover, in those papers, the interested reader can find additional pointers to validation studies using the same datasets. For completeness, it is worth reporting that they have also been used for benchmarking in the context of clustering analysis of microarray data [32, 62, 122, 139].

Particularly relevant is their use in two studies strictly related to this thesis: Giancarlo et al. [62] and Monti et al. [122]. Indeed, Giancarlo et al. use the following six datasets: CNS Rat, Leukemia, Lymphoma, NCI60, Yeast and PBM, while the remain three datasets are used by Monti et al.. The choice to use all nine is motivated as follows: the datasets of Giancarlo et al. allow to make an accurate and uniform benchmarking of several internal validation measures, taking into account both time and precision as well as to compare our results with extant ones in the Literature. This study is reported in Chapter 6 and it is quite unique in literature. The datasets of Monti et al. are used to complete the study of `Consensus` and to compare it with a speedup proposed in Chapter 7.

Although the Giancarlo et al.'s datasets have relatively few items to classify and relatively few dimensions, it is worth mentioning that Lymphoma, NCI60 and Leukemia have been obtained by Dudoit and Fridlyand and Handl et al., respectively, via an accurate statistical screening of the three relevant microarray experiments that involved thousands of conditions (columns). That screening process eliminated most of the conditions since there was no statistically significant variation across items (rows). Indeed, one would hardly attempt the clustering of microarray experiments without a preliminary statistical screening aimed at identifying the "relevant parts" of the experiment [51]. It is also worth pointing out that the three mentioned datasets are quite representative of microarray cancer studies. The CNS Rat and Yeast datasets come from gene functionality studies. The sixth one, PBM, is a dataset that corresponds to a cDNA with a large number of items to classify and it is used to show the current limitations of existing validation methods (see Giancarlo et al. for additional details). Indeed, they have been established with PBM as input. In particular, when given to `Consensus` as input, the computational demand is such that all experiments were stopped after four days, or they would have taken weeks to complete. It is also worth pointing out that the remain three datasets are very high dimension ($\simeq 1000$ features). Therefore, they naturally complement the first six since they all have relatively few features (less than 200). In summary, the nine datasets used for the experimentation in this dissertation seem to be a reliable sample of microarray studies, where clustering is used as an exploratory data analysis technique.

CNS Rat: It is a 112×17 data matrix, obtained from the expression levels of 112 genes during a rat's central nervous system development. The dataset is studied by Wen et al. [172], where they suggest a partition of the genes into six classes, four of which are composed of biologically, functionally-related genes. This partition is taken as the gold solution, which is the same one used for the validation of `FOM`.

Leukemia: It is a 38×100 data matrix, where each row corresponds to a patient with acute leukemia and each column to a gene. The original microarray experiment consists of a 72×6817 matrix, due to Golub et al. [65]. In order to obtain the current dataset, Handl et al. [73] extracted from it a 38×6817 matrix, corresponding to the "learning set" in the study of Golub et al. and, via preprocessing steps, they reduced it to the current dimension by excluding genes that exhibited no significant variation across samples. The interested reader can find details of the extraction process in Handl et al.. For this dataset, there is a partition into three classes and it is taken as gold solution. It is also worthy of mention that Leukemia has become a benchmark standard in the cancer classification community [30].

Lymphoma: It is a 80×100 data matrix, where each row corresponds to a tissue sample and each column to a gene. The dataset comes from the study of Alizadeh et al. [8] on the three most common adult lymphoma tumors. There is a partition into three classes and it is taken as the gold solution. The dataset has been obtained from the original microarray experiments, consisting of an 80×4682 data matrix, following the same preprocessing steps detailed in Dudoit and Fridlyand [48].

NCI60: It is a 57×200 data matrix, where each row corresponds to a cell line and each column to a gene. This dataset originates from a microarray study in gene expression variation among the sixty cell lines of the National Cancer Institute anti-cancer drug screen [2], which consists of a 61×5244 data matrix. There is a partition of the dataset into eight classes, for a total of 57 cell lines, and it is taken as the gold solution. The dataset has been obtained from the original microarray experiments as described by Dudoit and Fridlyand [48].

Normal: It is a 90×1277 data matrix, where each row corresponds to a tissue sample and each column to a gene. The dataset comes from the study of Su et al. [161] on four distinct cancer types. There is a partition into four classes and it is taken as the gold solution.

Novartis: It is a 103×1000 data matrix, where each row corresponds to a tissue sample and each column to a gene. The dataset comes from the study of Ramaswamy et al. [140] on 13 distinct tissue types. There is a partition into 13 classes and it is taken as the gold solution.

PBM: It is a 2329×139 data matrix, where each row corresponds to a cDNA with a fingerprint of 139 oligos. According to Hartuv et al. [78], the cDNAs in the dataset originated from 18 distinct genes, i.e., the classes are known. The partition of the dataset into 18 groups was obtained by lab experiments at Novartis in Vienna. Following that study, this partition is taken as the gold solution.

St.Jude: It is a 248×985 data matrix, where each row corresponds to a tissue sample and each column to a gene. The dataset comes from the study of Yeoh et al. [179] on diagnostic bone marrow samples from pediatric acute leukemia patients corresponding to 6 prognostically important leukemia sub-types. There is a partition into 6 classes and it is taken as the gold solution.

Yeast: It is a 698×72 data matrix, studied by Spellman et al. [159] whose analysis suggests a partition of the genes into five functionally-related classes which is taken as the gold solution and which has been used by Shamir and Sharan for a case study on the performance of clustering algorithms [152].

In this dissertation, it is referred to as `Benchmark 1` the following group of datasets: CNS Rat, Leukemia, Lymphoma, NCI60, PBM and Yeast. While the remaining three datasets are referred to as `Benchmark 2`.

5.1.2 Simulated Data

In order to compare some of the algorithms of this thesis, in particular in the speedup of `Consensus` proposed in Chapter 7, with the work of Monti et al. [122] on `Consensus`, some artificial datasets from that study are used. These datasets have known characteristics, typical of microarray data, but since they have no “noise”, they are “easy” to classify. Therefore, the experimental results involving them are considered as complementary to those on real microarray datasets. The three datasets, together with the acronyms used in this dissertation, are reported next.

Gaussian3: It is a 60×600 data matrix. It is generated by having 200 distinct features out of the 600 assigned to each cluster. There is a partition into three classes and that is taken as the gold solution. The data simulates a pattern whereby a distinct set of 200 genes is up-regulated in one of the three clusters, and down-regulated in the remaining two.

Gaussian5: It is a 500×2 data matrix, and it is show in Fig. 5.1. It represents the union of observations from 5 bivariate Gaussians, 4 of which are centered at the corners of the square of side length λ, with the 5th Gaussian centered at $(\lambda/2, \lambda/2)$. A total of 250 samples, 50 per class, were generated, where two values of λ are used, namely, $\lambda = 2$ and $\lambda = 3$, to investigate different levels of overlapping between clusters. There is a partition into five classes and that is taken as the gold solution.

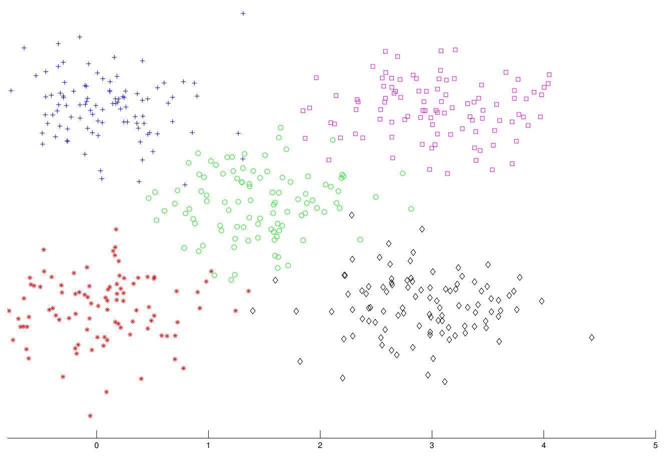

Figure 5.1: The Gaussian5 dataset.

Simulated6: It is a 60×600 data matrix. It consists of a 600-gene by 60-sample dataset. It can be partitioned into 6 classes with 8, 12, 10, 15, 5, and 10 samples respectively, each marked by 50 distinct genes uniquely up-regulated for that class. Additionally, 300 noise genes (i.e., genes having the same distribution within all clusters) are included. The genes for the different clusters are of varying "sharpness". That is, the 50 genes marking the first class are the sharpest- with highest differential expression and lowest variation-followed by the 50 genes for the second cluster, etc. Fig. 5.2 depicts the expression profile of the 600 genes within each cluster. This partition into 6 classes is taken as the gold solution.

This three datasets are also included in the `Benchmark 2`.

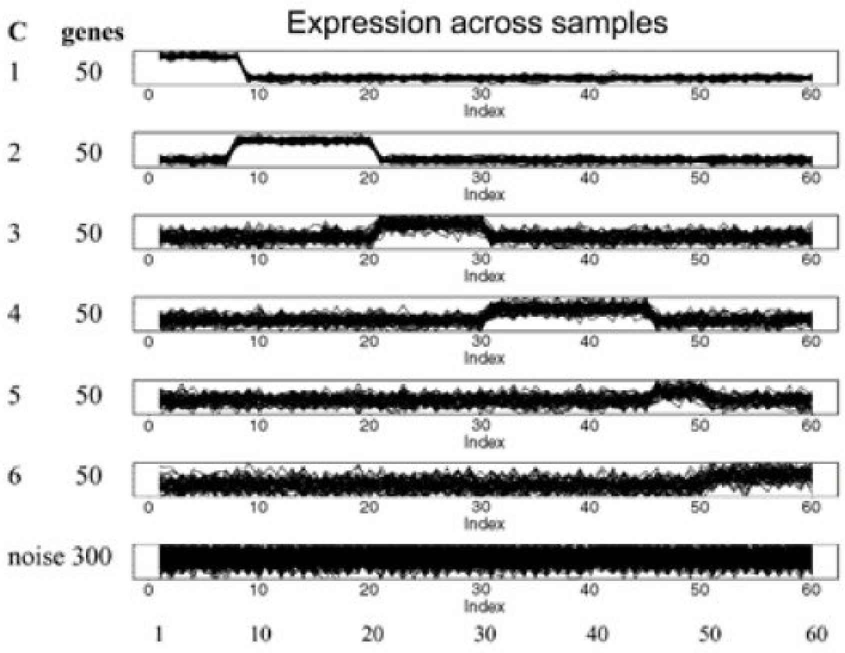

Figure 5.2: Expression profiles for each gene within each cluster on the Simulated6 dataset. This figure is takes from [122]

5.2 Clustering Algorithms and Their Stability

In this dissertation, a suite of clustering algorithms is used. Among the hierarchical methods [87] Hier-A (Average Link), Hier-C (Complete Link), and Hier-S (Single Link) (see Section 1.2.1).

Moreover, both K-means [87] and NMF are used (see Section 1.2.2 and Chapter 4) , both in the version that starts the clustering from a random partition of the data and in the version where each takes, as part of its input, an initial partition produced by one of the chosen hierarchical methods. For K-means, the acronyms of those versions are K-means-R, K-means-A, K-means-C and K-means-S, respectively. An analogous notation is followed for NMF.

It is worth pointing out that K-means-R is a randomized algorithm that may provide different answers on the same input dataset. That might make the values of many of the measures studied in this dissertation to depend critically on the particular execution of the algorithm. Such a dependance is important for WCSS, KL and FOM. For those measures and their approximations, the computation of the relevant curves, on all datasets, with K-means-R is repeated five times. Only negligible differences from run to run is observed. Therefore, in what follows, all reported results refer to a single run of the algorithms, except for the cases in which an explicit Monte Carlo simulation is required.

For completeness, it is also reported that in this thesis a C/C++ implementation of the NMF is used, which is based on the Matlab script available at the Broad institute [1]. Indeed, it was converted to a C/C++ version that was then validated by ensuring it produced the same results as for the Matlab version, in a number of simulations. Notice that this implementation also allows for NMF to start from two matrices W and H that actually correspond to a partition of the data into clusters, rather than choosing W and H at random. Such an option is analogous to the well known one offered by K-means, which can start from a given solution rather than randomly. As with K-means, `NMF` also has a faster convergence to a solution when not initialized at random, although the improvement seems not to be significant.

5.3 Similarity/Distance Functions

All of the algorithms use Euclidean distance in order to assess similarity of single elements to be clustered. Such a choice is natural and conservative, as now explained. It places all algorithms in the same position without introducing biases due to distance function performance, rather than to the algorithm. Moreover, time course data have been properly standardized (mean equal to zero and variance equal to one), so that Euclidean distance would not be penalized on those data. This is standard procedure, e.g., [181], for those data. The results obtained are conservative since, assuming that one has a provably much better similarity/distance function, one can only hope to get better estimates than ours (else the used distance function is not better than Euclidean distance after all). As it is clear from the upcoming results presented in the next chapters, such better estimates will cause no dramatic change in the general picture of our findings. The choice is also natural, in view of the debate regarding the identification of a proper similarity/distance function for clustering gene expression data and the number of such measures available. The state of the art as well some relevant progress in the identification of such measure is well presented in [60, 61, 139].

5.4 Hardware

All experiments for the assessment of the precision of each measure were performed in part on several state-of-the-art PCs and in part on a 64-bit AMD Athlon 2.2 GHz bi-processor with 1 GB of main memory running Windows Server 2003. All the timing experiments reported were performed on the bi-processor, using one processor per run. The usage of different machines for the experimentation was deemed necessary in order to complete the full set of experiments in a reasonable amount of time. Indeed, as detailed later, some measures require weeks to complete execution on large datasets. It is worth pointing out that all the Operating Systems supervising the computations have a 32 bits precision.

5.5 NMF Benchmarking

In this section a benchmarking of NMF as a clustering algorithms is described. In order to perform it, the performance of NMF is measured via the three external indices described in Section 2.1: `Adjusted Rand` Index, `FM-Index` and `F-Index`.

External indices can be very useful in evaluating the performance of algorithms and internal/relative indices, with the use of datasets that have a gold standard solution. A brief illustration is given of the methodology for the external validation of a clustering algorithm, via an external index that needs to be maximized. The same methodology applies to internal/relative indices, as discussed in [181]. For a given dataset, one plots the values of the index computed by the algorithm as a function of k, the number of clusters. Then, one expects the curve to grow to reach its maximum close or at the number of classes in the reference classification of the dataset. After that number, the curve should fall.

	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	875	219	500	921	594	4.4×10^5
Hier-C	865	250	469	750	625	4.6×10^5
Hier-S	860	296	516	641	609	4.3×10^5
K-means-R	3.2×10^3	2.1×10^3	3.2×10^3	7.2×10^3	1.1×10^5	1.1×10^6
K-means-A	3.2×10^3	1.1×10^3	4.2×10^3	4.4×10^3	1.1×10^5	1.7×10^6
K-means-C	2.9×10^3	1.1×10^3	4.0×10^3	4.2×10^3	1.0×10^5	1.3×10^6
K-means-S	3.3×10^3	1.3×10^3	5.2×10^3	5.4×10^3	1.2×10^5	1.4×10^6
NMF-R	9.0×10^6	8.6×10^4	3.9×10^5	5.2×10^5	2.9×10^8	-
NMF-A	3.0×10^6	2.4×10^4	7.9×10^4	1.1×10^5	5.5×10^7	-
NMF-C	2.4×10^6	2.5×10^4	7.4×10^4	1.1×10^5	5.9×10^7	-
NMF-S	5.7×10^6	2.3×10^4	6.9×10^4	1.1×10^5	4.5×10^7	-

Table 5.1: Time results in millisecond for all the algorithms on `Benchmark 1` datasets. For PBM, the experiments were terminated due to their high computational demand (weeks to complete).

In what follows, the results of the experiments are presented, with the use of the indices. The experiments summarized here refer to the `Benchmark 1` datasets and the simulated datasets in `Benchmark 2`. For each dataset and each clustering algorithm, each index is computed for a number of cluster values in the range $[2, 30]$. Moreover, the time performance of NMF on the microarray datasets is compared with that of the classical clustering algorithms. To this end the execution time in millisecond of each algorithm on each datasets is reported in Table 5.1. A dash indicates that the experiment was stopped because of its high computational demand. Indeed, given the dimension of the PBM datasets, NMF is stopped after four days, for this reason the results on this dataset are not reported here. From the results in Table 5.1, it is possible to see that NMF is very slow, at least four order of magnitude of difference with the other clustering algorithms. Moreover, NMF is not able to complete the experiment on PBM dataset.

Adjusted Rand Index For those experiments, the relevant plots are in Fig. 5.3-5.4 for `Benchmark1` and simulated datasets, respectively. Based on the results on `Benchmark1` datasets (see Fig. 5.3), it is possible to see that all the algorithms perform very well on Leukemia and NCI60 datasets. For the remain three datasets, their performance is somewhat mixed, and sometimes they are not precise. In particular, the Hier-S, NMF-R, and NMF-S algorithms. From the results on the simulated datasets (see Fig. 5.4), it is possible to see that, except for Hier-S and K-means-S, all the algorithms perform very well on Gaussian3 and Gaussian5. Whereas, on Simulated6 all the algorithms give an useless indication.

FM-Index For those experiments, the relevant plots are in Fig. 5.5-5.6. Based on them, Hier-S, NMF-S are still the worst among the algorithms, however now there is no consistent indication given by the other algorithms. Moreover, on Gaussian5 and Simulated6 NMF, both with random and hierarchical initialization, does not perform very well.

F-Index For those experiments, the relevant plots are in Figs. 5.7-5.8. On `Benchmark1` datasets, Hier-S and NMF-S are still the worst among the algorithms and the indications in this case about the other algorithms are essentially the same as in the case of `Adjusted Rand` Index. Whereas, on simulated dataset all the algorithms have a disappointing performance.

In Conclusion, from all the results proposed in this section, it should be mentioned that although all of the three indices have solid statistical justifications, the `Adjusted Rand` Index seems to be the best performer while the `FM-index` is somewhat disappointing. Moreover, the use of NMF as a clustering algorithm is not suggested, in particular for large datasets. Indeed, given the steep computational price (see Table 5.1) one has to afford, its use does not seem to be justified since Hier-A is at least four orders of magnitude faster and with a better precision. In fact, the main power of NMF rests on its pattern discovery ability, and its use as a clustering algorithm seems to be very limiting for this technique.

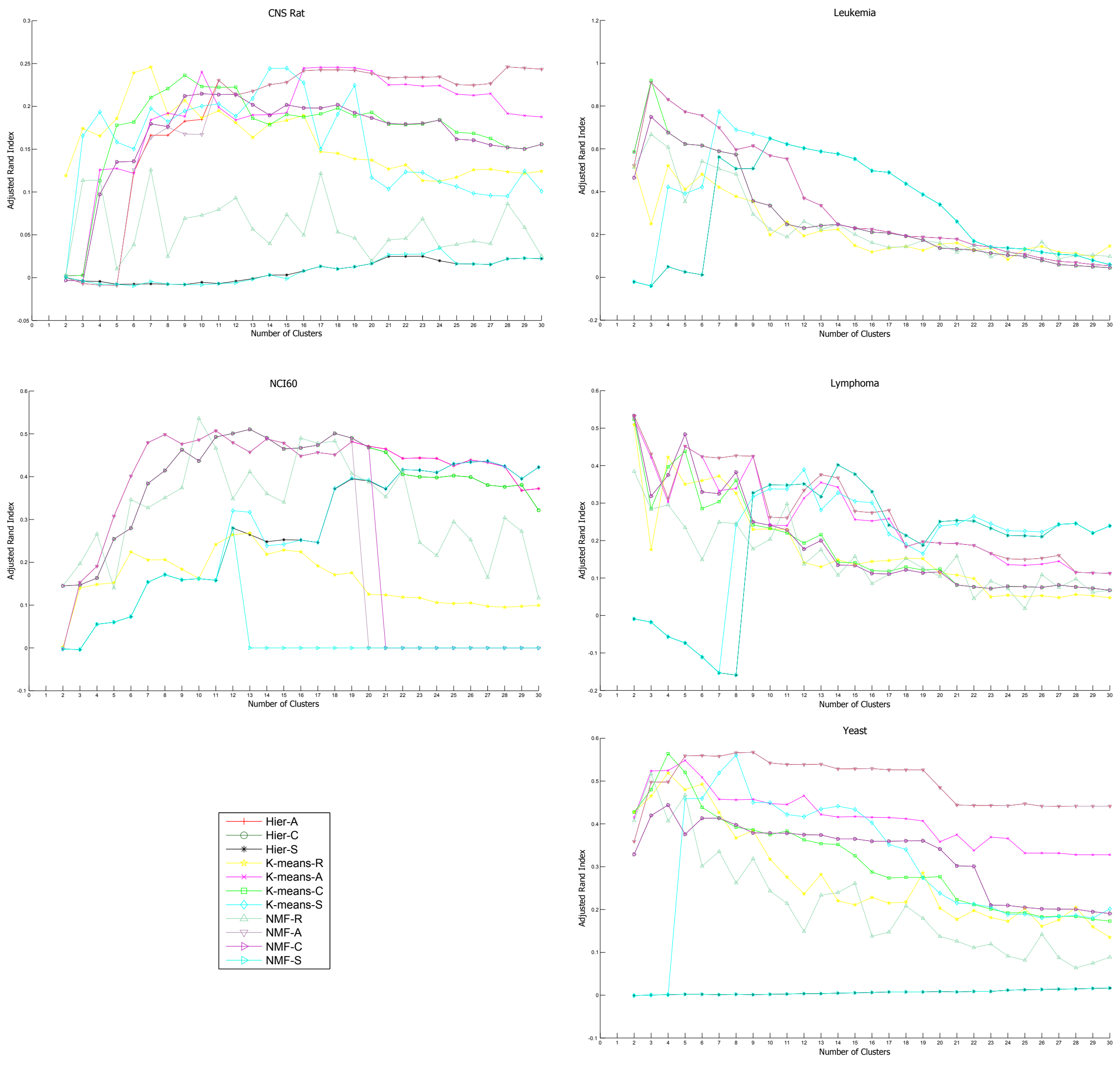

Figure 5.3: The `Adjusted Rand` Index curves, for each of the `Benchmark1` datasets. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm. For PBM, the experiments on NMF were terminated due to their high computational demand and the corresponding plots has been removed from the figure.

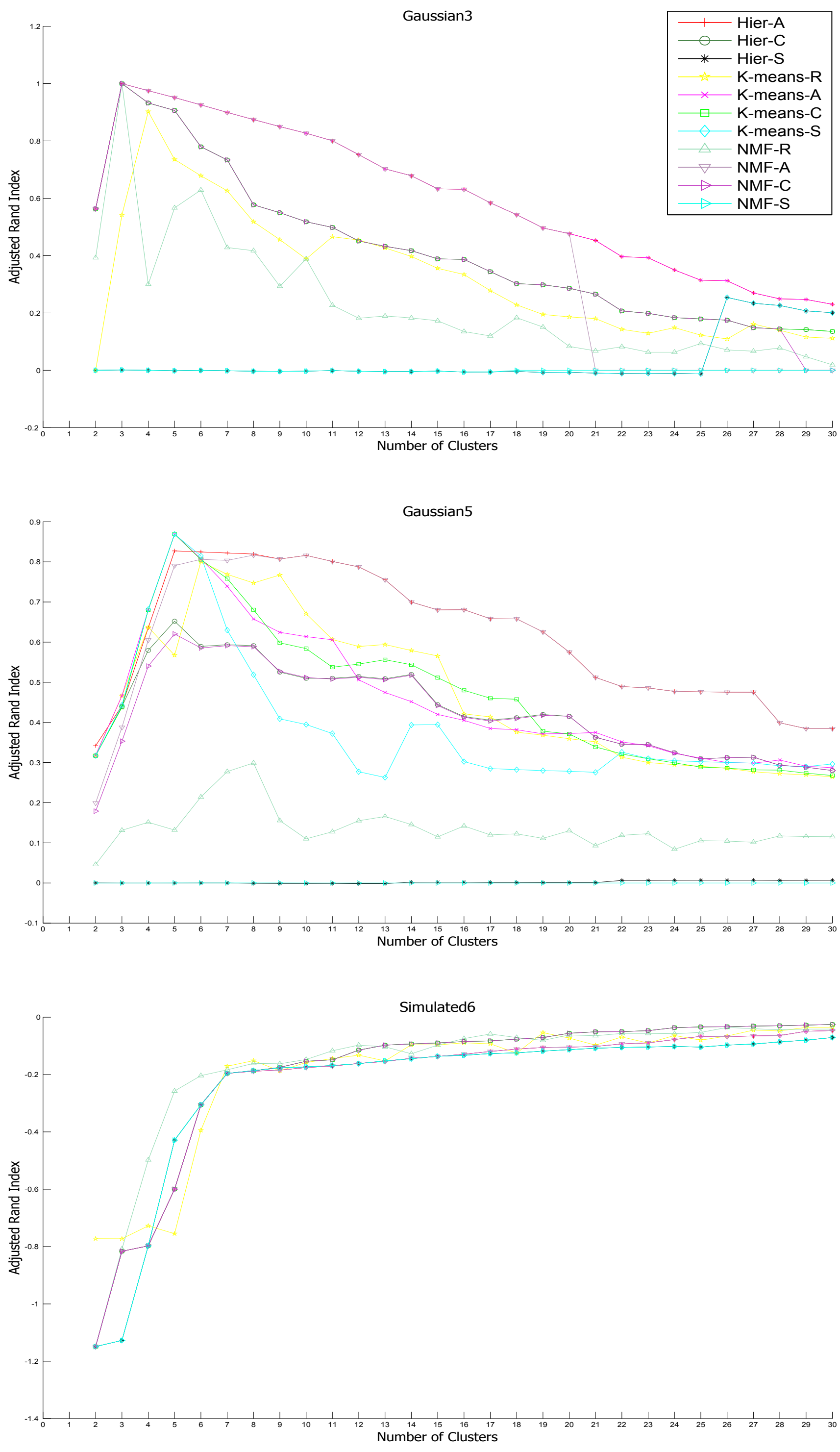

Figure 5.4: The `Adjusted Rand` Index curves, for each of the simulated dataset in `Benchmark 2`. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm.

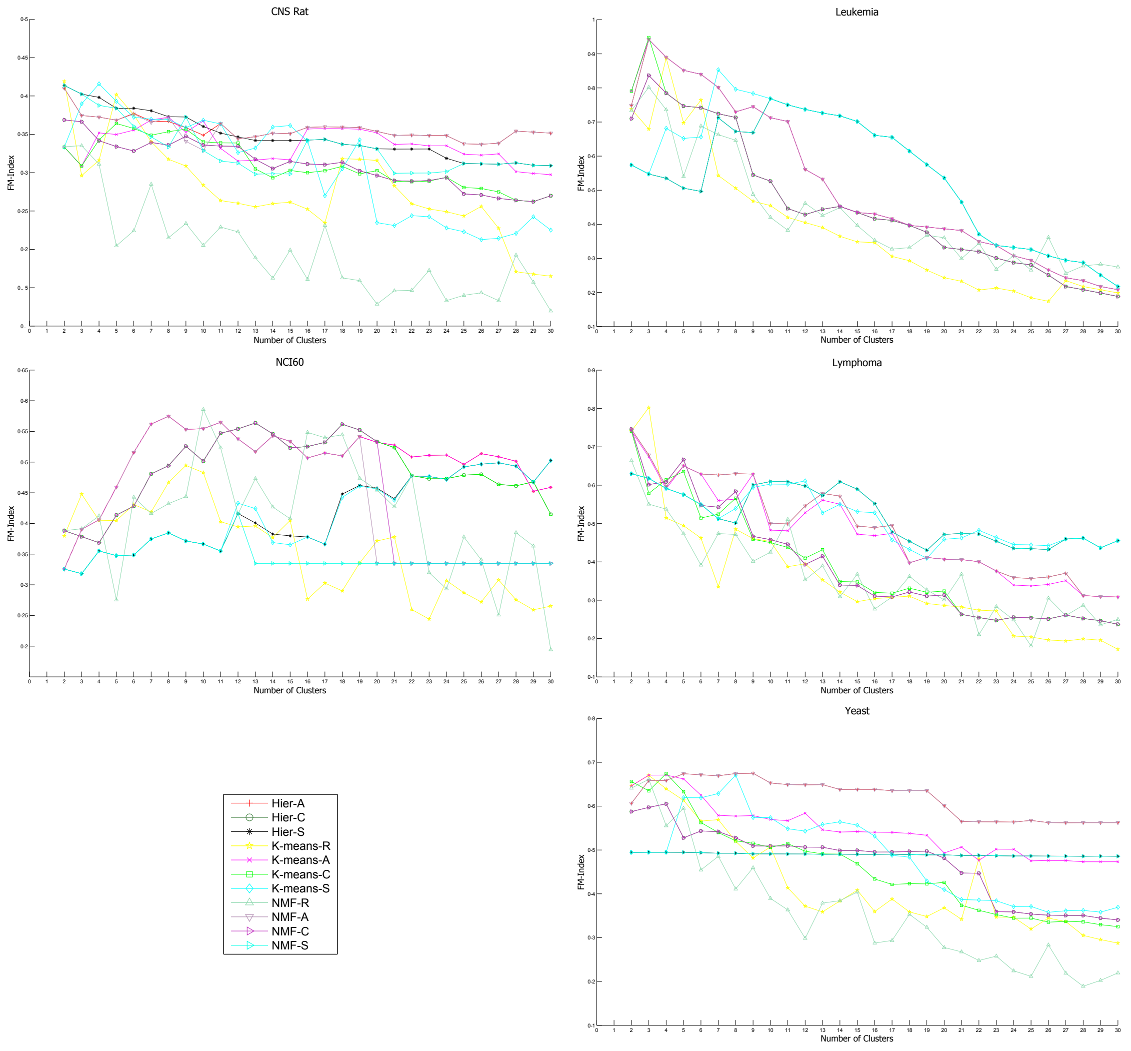

Figure 5.5: The `FM-Index` curves, for each of the `Benchmark1` datasets. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm. For PBM, the experiments on NMF were terminated due to their high computational demand and the corresponding plots has been removed from the figure.

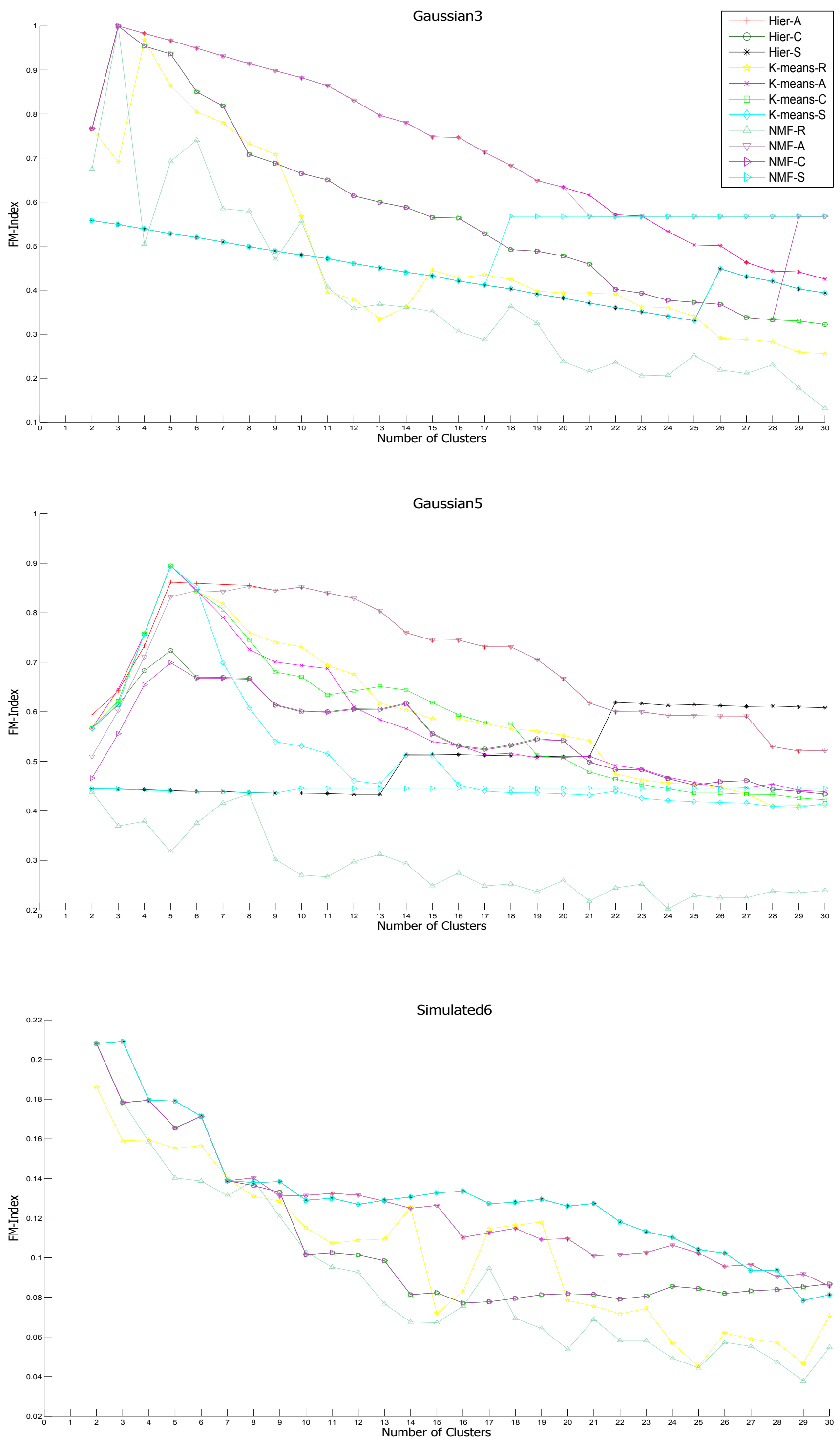

Figure 5.6: The `FM-index` curves, for each of the simulated dataset in `Benchmark 2`. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm.

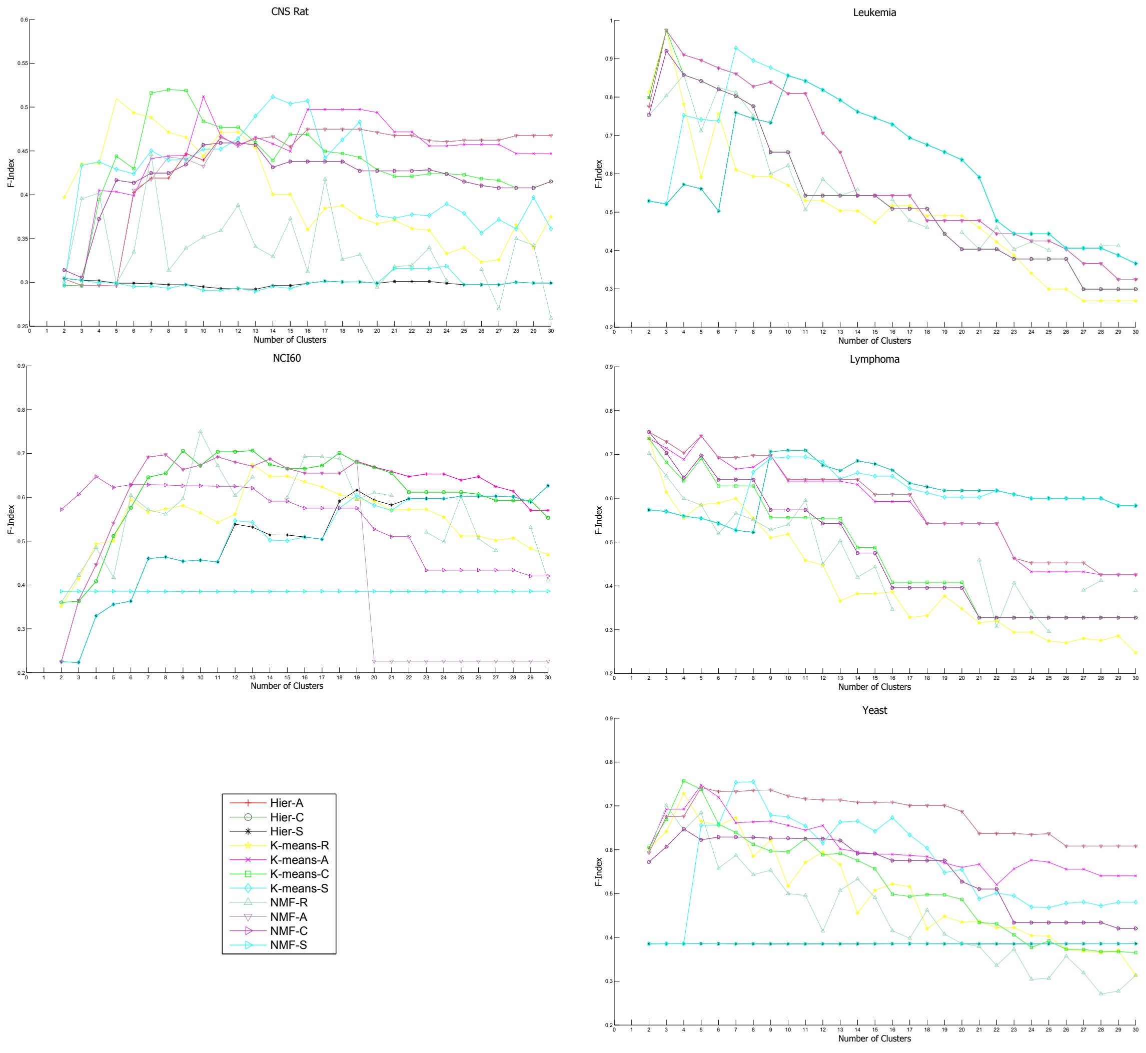

Figure 5.7: The `F-Index` curves, for each of the `Benchmark1` datasets. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm. For PBM, the experiments on NMF were terminated due to their high computational demand and the corresponding plots has been removed from the figure.

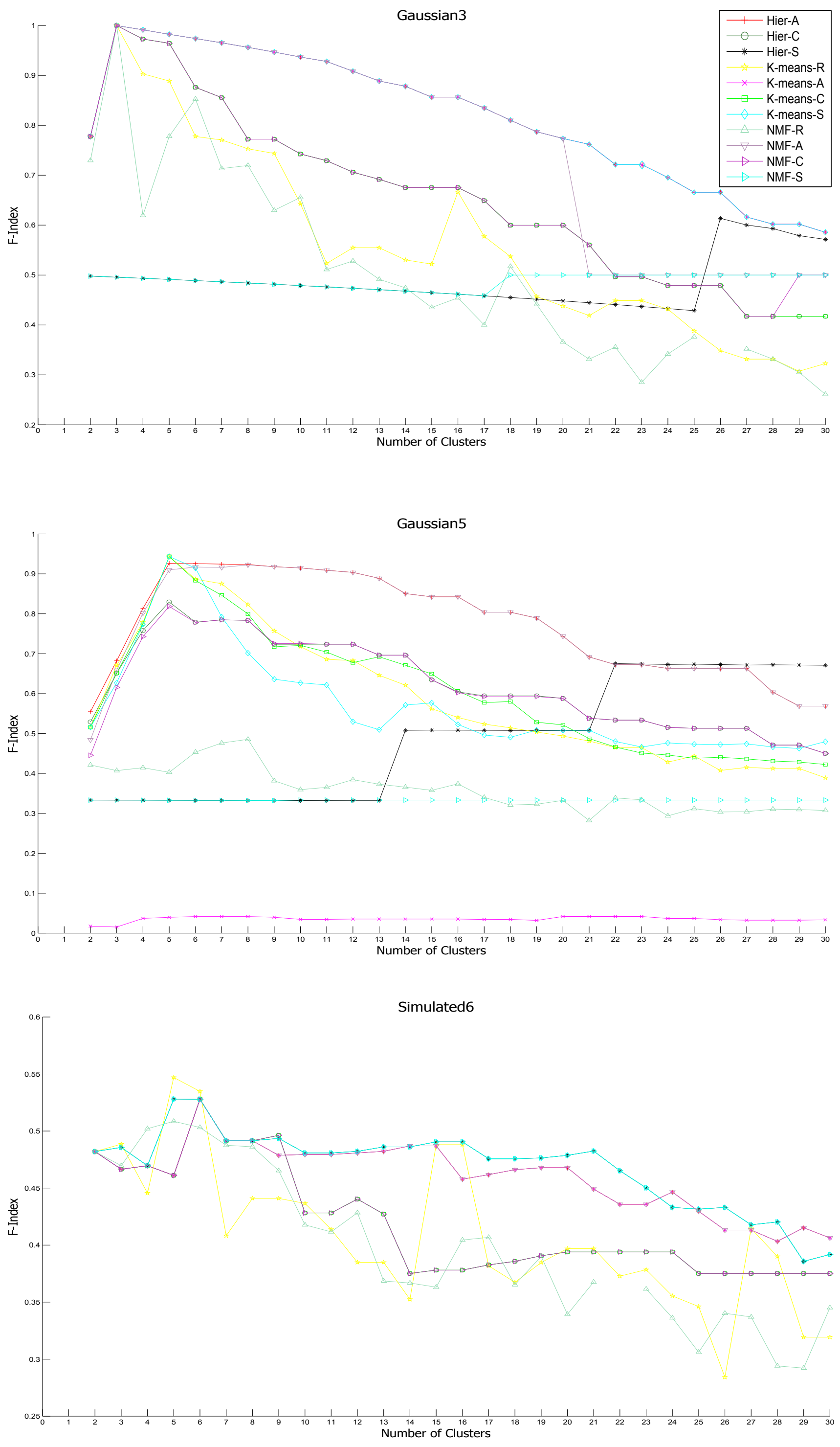

Figure 5.8: The `F-Index` curves, for each of the simulated dataset in `Benchmark 2`. In each figure, the plot of the index, as a function of the number of clusters, is plotted differently for each algorithm.

Chapter 6

An Experimental Assessment of Internal Validation Measures

In this chapter, a benchmarking of some of the internal validation measures described in the previous chapters is presented. In particular, all measures presented in Chapter 2 are considered, while only `ME`, `Consensus` and `Clest` are studied here, since they seem to be the most representative of the stability based ones. This study tries to establish the intrinsic, as well as the relative, merit of a measure taking into account both its predictive power and its computational demand. To the best of our knowledge, this is the first study of this kind available in the Literature. It is worthy to anticipate that, based on the results reported here, a speedup of some of the measure presented here becomes a very natural and well motivated problem, that is addressed in the next chapter.

6.1 Problems Statement and State of the Art

An established and rich research area in bioinformatics is the design of new internal validation measures that should assess the biological relevance of the clustering solutions found. Despite the vast amount of knowledge available in this area in the general data mining literature [52, 75, 77, 79, 88, 93, 121, 144], gene expression data provide unique challenges. Indeed, the internal validation measure must predict how many clusters are really present in a dataset, an already difficult task, made even worse by the fact that the estimation must be sensible enough to capture the inherent biological structure of functionally related genes. The excellent survey by Handl et al. [73] is a big step forward in making the study of internal validation measures a central part of both research and practice in bioinformatics, since it provides both a technical presentation as well as valuable general guidelines about their use for post-genomic data analysis. Although much remains to be done, it is, nevertheless, an initial step.

In order to establish the intrinsic and relative merit of a measure, the two relevant questions are:

(A) What is the precision of a measure, i.e., its ability to predict the correct number of clusters in a dataset? That is usually established by comparing the number of clusters predicted by the measure against the number of clusters in the gold solution of several datasets. It is worth recalling from Chapter 5 that the gold solution is a partition of the dataset in classes that can be trusted to be correct, i.e., distinct groups of functionally related genes.

(B) Among a collection of measures, which is more accurate, less algorithm-dependent, etc.,?. Precision versus the use of computational resources, primarily execution time, would be an important discriminating factor.

Although the classic studies in the general data mining Literature, mentioned earlier, are also of great relevance for bioinformatics, there is an acute need for analogous studies conducted on internal measures introduced recently and specifically designed for analysis of microarray data. In this chapter both of the stated questions are addressed for several measures. They are all characterized by the fact that, for their prediction, they make use of nothing more than the dataset available (see Chapters 2 and 3): `WCSS`, `KL`, `Clest`, `Consensus` , `FOM`, `Gap` and `ME`. In order to perform this study, as anticipated in Chapter 5, only the `Benchmark1` datasets is used, i.e.: CNS Rat, Leukemia, Lymphoma, NCI60, Yeast and PBM.

Initial studies of the mentioned measures, in connection with both Questions (A) and (B), have been done, primarily, in the papers in which they were originally proposed. This study carries further those studies by providing more focused information about using those measures for the analysis of gene expression data. For Question (A), that analysis provides further insights into the properties of the mentioned measures, with particular attention to time. For Question (B), a first comparative analysis involving all of those measures that accounts for both precision and time is provided. This is particularly relevant in regard to the "stability-based" methods, i.e., `Clest`, `Consensus` and `ME`. In fact,

(1) those three measures are excellent representatives of methods in the class (see Handl et al. and [92, 116]);

(2) Dudoit and Fridlyand mention that it would be desirable to relate `Clest` and `ME` but no comparison seems to be available in the literature;

(3) although it is quite common to include `Clest` and `Gap` in comparative analysis for novel measures, `Consensus` is hardly considered. However, the experiments presented here show that it should definitely be included.

Finally, it is worth pointing out that the results and conclusion of this chapter are also available to Giancarlo et al. [62].

6.2 Intrinsic Precision of the Internal Measures

In this section, the experiments with the aim to shed some light on Question (A) are presented. As discussed in Chapters 2 and 3, for most measures, the prediction of the "optimal" number k^* of clusters is based on the visual inspection of curves and histograms. For conciseness, all the relevant material is provided in the following supplementary material web site [3] (Figures section). Here only summary tables are given, based on the corresponding analysis of the relevant curves and experiments. In this section, two separate tables for each measure are reported, one for the precision and the other for timing results.

It is worthy to anticipate that the next section addresses the relative merits of each measure and two global summary tables are reported, but only for the best performers. That is, for each measure, the experimental parameters are reported (e.g., clustering algorithm) only if in that setting the prediction of k^* has been reasonably close to the gold solution (at most an absolute value difference of one between the predicted number and the real number) in at least four of the six datasets used in this chapter.

Moreover, in what follows, for each cell in a table displaying precision results, a number in a circle with a black background indicates a prediction in agreement with the number

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	10	❸	3	6	❺	-
Hier-C	10	❸	⑦	8	9	-
Hier-S	8	10	⑦	9	-	-
K-means-R	4	❸	3	8	④	-
K-means-A	4	❸	⑦	6	❺	-
K-means-C	⑤	❸	❽	8	④	-
K-means-S	3	④	⑦	8	24	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 6.1: A summary of the precision results for `WCSS` on all algorithms and `Benchmark 1` datasets. Cells with a dash indicate that `WCSS` did not give any useful indication.

of classes in the dataset, while a number in a circle with a white background indicates a prediction that differs, in absolute value, by 1 from the number of classes in the dataset; when the prediction is one cluster, i.e. Gap statistics, this symbol rule is not applied because the prediction means no cluster structure in the data; a number not in a circle indicates the remaining predictions. As detailed in each table displaying timing or precision results, cells with a dash indicate that either the experiment was stopped, because of its high computational demand, or that the measure gives no useful indication. The timing results are reported only on the four smallest datasets. Indeed, for Yeast and PBM, the computational demand is such on some measures that either they had to be stopped or they took weeks to complete. For those two datasets, the experiments reported here were done using more than one machine.

6.2.1 WCSS

For each algorithm, and each dataset, `WCSS` is computed for a number of cluster values in the range $[2, 30]$. The relevant plots are in the Figures section at the following supplementary material web site [3]: Fig. S1 for the K-means algorithms and Fig. S2 for the hierarchical algorithms.

As outlined in the Section 2.2.1.1, given the relevant `WCSS` curve, k^* is predicted as the abscissa closest to the "knee" in that curve. The values resulting from the application of this methodology to the relevant plots are reported in Table 6.1, while the timing results for the relevant datasets are reported in Table 6.2.

One has that `WCSS` performs well with K-means-C and K-means-A (see Table 6.1), on the first five datasets, while it gives no reasonably correct indication on PBM. It is a poor performer with the other clustering algorithms. Those facts give strong indication that `WCSS` is algorithm-dependent. Finally, the failure of `WCSS`, with all algorithms, to give a good prediction for PBM indicates that `WCSS` may not be of any use on large datasets having a large number of clusters.

Overall, the best performer is K-means-C. The relative results are reported in Tables 6.15 and 6.16, for comparison with the performance of the other measures.

6.2.2 KL

Following the same experimental setup of `WCSS`, the `KL` measure is computed, for each dataset and each algorithm. The results, summarized in Tables 6.3 and 6.4, are rather

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
Hier-A	1.1×10^3	4.0×10^2	2.1×10^3	1.9×10^3
Hier-C	7.0×10^2	4.0×10^2	1.7×10^3	1.4×10^3
Hier-S	2.6×10^3	6.0×10^2	3.2×10^3	3.8×10^3
K-means-R	2.4×10^3	2.0×10^3	8.4×10^3	8.4×10^3
K-means-A	2.3×10^3	1.3×10^3	5.4×10^3	5.8×10^3
K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
K-means-S	2.6×10^3	1.6×10^3	7.3×10^3	7.4×10^3

Table 6.2: A summary of the timing results for `WCSS`.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	3	②	17	12
Hier-C	10	❸	2	②	16	15
Hier-S	21	7	⑦	9	15	25
K-means-R	4	27	3	22	29	24
K-means-A	25	❸	3	②	7	16
K-means-C	2	❸	⑦	②	26	24
K-means-S	4	④	12	8	13	16
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 6.3: A summary of the precision results for `KL` on all algorithms and on `Benchmark 1` datasets.

disappointing: the measure provides some reliable indication, across algorithms, only on the Leukemia and the Lymphoma datasets. Due to such a poor performance, no results are reported in Tables 6.15 and 6.16, for comparison with the performance of the other measures.

6.2.3 Gap

For each dataset and each clustering algorithm, three versions of `Gap` are computed, namely `Gap-Ps`, `Gap-Pc` and `Gap-Pr`, for a number of cluster values in the range $[1, 30]$. `Gap-Ps` uses the Poisson null model, `Gap-Pc` the Poisson null model aligned with the principal

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
Hier-A	1.9×10^3	6.0×10^2	2.1×10^3	2.5×10^3
Hier-C	1.6×10^3	1.1×10^3	2.5×10^3	2.1×10^3
Hier-S	3.4×10^3	1.3×10^3	3.7×10^3	4.9×10^3
K-means-R	2.7×10^3	3.4×10^3	9.3×10^3	9.0×10^3
K-means-A	2.3×10^3	2.4×10^3	5.7×10^3	6.2×10^3
K-means-C	3.0×10^3	2.6×10^3	5.0×10^3	5.8×10^3
K-means-S	4.0×10^3	2.9×10^3	8.0×10^3	8.5×10^3

Table 6.4: A summary of the timing results `KL` on all algorithms.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
`Gap-Ps`-Hier-A	1	④	1	6	3	1
`Gap-Ps`-Hier-C	1 or 2	④	2	1 or 25	7	15
`Gap-Ps`-Hier-S	1	1	1	1	1	1
`Gap-Ps`-K-means-R	❻ or ⑦	④ or 5	3	8	9	7
`Gap-Ps`-K-means-A	⑦	❸	1	8	7	9
`Gap-Ps`-K-means-C	⑦	④	2	1 or 25	12	6
`Gap-Ps`-K-means-S	9	❸	1	1	7	8
`Gap-Pc`-Hier-A	1	❸ or ④	1	1	1 or 2 or 3	-
`Gap-Pc`-Hier-C	1	④	1	1	3	-
`Gap-Pc`-Hier-S	1	1	1	1	1	-
`Gap-Pc`-K-means-R	2	1	1	1	④	-
`Gap-Pc`-K-means-A	2	④	1	1	3	-
`Gap-Pc`-K-means-C	2	1	1	1	④	-
`Gap-Pc`-K-means-S	3	1	1	1	1	-
`Gap-Pr`-Hier-A	3	④	1	6	3	1
`Gap-Pr`-Hier-C	⑦	④	1	1 or 25	16	1
`Gap-Pr`-Hier-S	1 or ❻	1	2	1	1	2
`Gap-Pr`-K-means-R	❻	④	5	8	8	8
`Gap-Pr`-K-means-A	8	④	1	8	13	4
`Gap-Pr`-K-means-C	⑤	6	1	1 or 25	8	1
`Gap-Pr`-K-means-S	⑦	❸	2	1	11	1
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 6.5: A summary of the precision results for `Gap` on all algorithms and `Benchmark 1` datasets, with use of three null models. For `Gap-Pc`, on PBM, the experiments were stopped due to their high computational demand.

components of the data while `Gap-Pr` uses the permutational null model (see Section 1.4). For each of them, a Monte Carlo simulation is performed, 20 steps, in which the measure returns an estimated number of clusters for each step. Each simulation step is based on the generation of 10 data matrices from the null model used by the measure. At the end of each Monte Carlo simulation, the number with the majority of estimates is taken as the predicted number of clusters. Occasionally, there are ties and both numbers are reported. The relevant histograms are displayed at the following supplementary material web site [3] (Figures section): Figs. S3-S8 for `Gap-Ps`, Figs. S9-S13 for `Gap-Pc` and Figs. S14-S19 for `Gap-Pr`. The results are summarized in Tables 6.5 and 6.6. For PBM and `Gap-Pc`, each experiment was terminated after a week, since no substantial progress was being made towards its completion.

The results for `Gap` are somewhat disappointing, as Table 6.5 shows. However, a few comments are in order, the first one regarding the null models. Tibshirani et al. find experimentally that, on simulated data, `Gap-Pc` is the clear winner over `Gap-Ps` (they did not consider `Gap-Pr`). The results reported here show that, as the dataset size increases, `Gap-Pc` incurs into a severe time performance degradation, due to the repeated data transformation step. Moreover, on the smaller datasets, no null model seems to have the edge. Some of the results are also somewhat puzzling. In particular, although the datasets have cluster structure, many algorithms return an estimate of $k^* = 1$, i.e., no cluster structure in the data. An analogous situation was reported by Monti et al.. In their study, `Gap-Ps` returned

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
Gap-Ps-Hier-A	2.7×10^5	1.4×10^5	6.1×10^5	6.4×10^5
Gap-Ps-Hier-C	2.3×10^5	1.1×10^4	3.4×10^5	3.2×10^5
Gap-Ps-Hier-S	6.1×10^5	1.9×10^5	1.1×10^6	1.4×10^6
Gap-Ps-K-means-R	8.4×10^5	5.0×10^5	1.1×10^6	1.0×10^6
Gap-Ps-K-means-A	6.1×10^5	4.7×10^5	1.1×10^6	1.0×10^6
Gap-Ps-K-means-C	6.0×10^5	6.1×10^5	8.8×10^5	7.6×10^5
Gap-Ps-K-means-S	9.1×10^5	6.5×10^5	2.1×10^6	1.8×10^6
Gap-Pc-Hier-A	3.2×10^5	3.7×10^5	8.1×10^5	6.1×10^5
Gap-Pc-Hier-C	1.9×10^5	1.9×10^5	7.1×10^5	5.8×10^5
Gap-Pc-Hier-S	7.7×10^5	3.1×10^5	1.4×10^6	1.3×10^6
Gap-Pc-K-means-R	4.9×10^5	8.0×10^5	1.8×10^6	1.8×10^6
Gap-Pc-K-means-A	3.8×10^5	7.0×10^5	1.3×10^6	1.4×10^6
Gap-Pc-K-means-C	4.1×10^5	5.8×10^5	1.3×10^6	1.2×10^6
Gap-Pc-K-means-S	6.5×10^5	7.6×10^5	2.4×10^6	2.0×10^6
Gap-Pr-Hier-A	2.5×10^5	1.6×10^5	3.3×10^5	3.8×10^5
Gap-Pr-Hier-C	1.3×10^5	1.4×10^5	3.2×10^5	3.7×10^5
Gap-Pr-Hier-S	6.8×10^5	1.9×10^5	1.1×10^6	1.4×10^6
Gap-Pr-K-means-R	8.6×10^5	5.4×10^5	1.5×10^6	9.4×10^5
Gap-Pr-K-means-A	7.4×10^5	5.0×10^5	8.7×10^5	1.0×10^6
Gap-Pr-K-means-C	6.7×10^5	4.6×10^5	8.6×10^5	1.0×10^6
Gap-Pr-K-means-S	1.2×10^6	5.4×10^5	1.8×10^6	2.3×10^6

Table 6.6: A summary of the timing results for Gap on all algorithms, with use of three null models. For Gap-Pc, on PBM, the experiments were stopped due to their high computational demand.

$k^* = 1$ on two artificial datasets. Fortunately, an analysis of the corresponding Gap curve showed that indeed the first maximum was at $k^* = 1$ but a local maximum was also present at the correct number of classes, in each dataset. An analogous analysis of the relevant Gap curves is also performed here to find that, in analogy with Monti et al., most curves show a local maximum at or very close to the number of classes in each dataset, following the maximum at $k^* = 1$. An example curve is given in Fig. 6.1. From the above, one can conclude that inspection of the Gap curves and *domain knowledge* can greatly help in disambiguating the case $k^* = 1$. It is worth pointing out that experiments conducted by Dudoit and Fridlyand and, independently by Yan and Ye [178], show that Gap tends to overestimate the correct number of clusters, although this does not seem to be the case for the datasets and algorithms used in this dissertation. The above considerations seem to suggest that the automatic rule for the prediction of k^* based on Gap is rather weak.

6.2.4 Clest

For CNS Rat and Yeast and each clustering algorithm, Clest is computed for a number of cluster values in the range $[2, 30]$ while, for Leukemia, NCI60 and Lymphoma, the ranges $[2, 10]$, $[2, 15]$ and $[2, 15]$ are used, respectively, due to the small size of the datasets. Moreover, although experiments have been started with PBM, no substantial progress was made after a week of execution and, for each clustering algorithm, the corresponding experiment was terminated. Following the same experimental setup of Dudoit and Fridlyand, for each cluster value k, 20 resampling steps and 20 iterations are performed. In each step, 66% of the rows of the data matrix are extracted, uniformly and at random, to create a learning set, to be given to the clustering algorithm to be clustered in k groups. As one of its input parameters, Clest requires the use of an external index E to establish the level of agreement between two partitions of a dataset. Here each of the following are used: the FM (the FM-Index), Adj (the Adjusted Rand Index) and F (the F-Index) (see Section 2.1).

The precision results are summarized in Table 6.7, while the timing results are reported in Table 6.8. The Leukemia, NCI60 and Lymphoma datasets were excluded since the experiments were performed on a smaller interval of cluster values with respect to CNS Rat. This latter interval is the standard one used in this dissertation to make consistent comparisons across measures and algorithms.

The results show that Clest has severe time demand limitations on large datasets. It also seems to achieve a better performance, across algorithms with Adj and F. Moreover, it is clearly algorithm-dependent, with K-means-R being the best performer with both FM and F. Those results are reported in Tables 6.15 and 6.16 for comparison with the performance of the other measures.

6.2.5 ME

For each of the first five datasets and each clustering algorithm, ME is computed for a number of cluster values in the range $[2, 30]$. Following the same experimental setup of Ben-Hur et al., for each cluster value k, 100 iterations are performed. In each step, two datasets to be given to the algorithm to be clustered in k groups are computed. Each dataset is created by extracting uniformly and at random 80% of the rows. The prediction of k^* is based on the plot of the corresponding histograms, as illustrated in Chapter 3. As for the external indices that are used, they are the same three used for Clest. The histograms obtained from such an experimentation are reported at the following supplementary material web site [3] in Figs. S20-S124. As for PBM, the computations were stopped because of their computational demand. A summary of the results is given in Tables 6.9 and 6.10. Indeed, the performance

	Precision				
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
Clest-FM-Hier-A	10	6	10	13	24
Clest-FM-Hier-C	10	④	⑨	15	8
Clest-FM-Hier-S	20	10	15	15	1
Clest-FM-K-means-R	8	④	❽	②	④
Clest-FM-K-means-A	18	7	12	15	13
Clest-FM-K-means-C	12	5	12	11	④
Clest-FM-K-means-S	24	8	13	15	1
Clest-Adj-Hier-A	13	❸	3	②	11
Clest-Adj-Hier-C	9	④	2	②	④
Clest-Adj-Hier-S	4	7	⑨	7	26
Clest-Adj-K-means-R	⑤	④	3	②	2
Clest-Adj-K-means-A	12	❸	3	②	❺
Clest-Adj-K-means-C	9	②	2	②	④
Clest-Adj-K-means-S	20	6	13	6	10
Clest-F-Hier-A	⑦	7	10	15	27
Clest-F-Hier-C	9	❸	13	❸	❺
Clest-F-Hier-S	28	10	15	15	1
Clest-F-K-means-R	❻	❸	15	②	④
Clest-F-K-means-A	8	6	10	14	11
Clest-F-K-means-C	9	5	12	❸	④
Clest-F-K-means-S	21	10	15	15	1
Gold solution	**6**	**3**	**8**	**3**	**5**

Table 6.7: A summary of the precision results for Clest on all algorithms and the first four datasets, with use of three external indices. For PBM, the experiments were terminated due to their high computational demand (weeks to complete).

	Timing
	CNS Rat
`Clest-FM`-Hier-A	1.1×10^6
`Clest-FM`-Hier-C	1.1×10^6
`Clest-FM`-Hier-S	1.1×10^6
`Clest-FM`-K-means-R	1.2×10^6
`Clest-FM`-K-means-A	1.4×10^6
`Clest-FM`-K-means-C	1.5×10^6
`Clest-FM`-K-means-S	1.8×10^6
`Clest-Adj`-Hier-A	1.1×10^6
`Clest-Adj`-Hier-C	1.1×10^6
`Clest-Adj`-Hier-S	1.1×10^6
`Clest-Adj`-K-means-R	1.1×10^6
`Clest-Adj`-K-means-A	1.4×10^6
`Clest-Adj`-K-means-C	1.4×10^6
`Clest-Adj`-K-means-S	1.8×10^6
`Clest-F`-Hier-A	1.1×10^6
`Clest-F`-Hier-C	1.1×10^6
`Clest-F`-Hier-S	1.1×10^6
`Clest-F`-K-means-R	1.2×10^6
`Clest-F`-K-means-A	1.4×10^6
`Clest-F`-K-means-C	1.5×10^6
`Clest-F`-K-means-S	1.8×10^6

Table 6.8: A summary of the timing results for `Clest` on all algorithms, with use of three external indices. For PBM, the experiments were terminated due to their high computational demand (weeks to complete). Therefore, the resulting column is omitted from the table. For the Leukemia, NCI60 and Lymphoma datasets, the timing experiments are not reported because incomparable with those of CNS Rat and of the other measures. The corresponding columns are eliminated from the table.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
ME-FM-Hier-A	4	②	2	②	1	-
ME-FM-Hier-C	2	②	2	②	1	-
ME-FM-Hier-S	8	②	2	②	-	-
ME-FM-K-means-R	2	②	2	②	3	-
ME-FM-K-means-A	2	②	4	②	2	-
ME-FM-K-means-C	2	②	2	②	3	-
ME-FM-K-means-S	2	②	3	②	④	-
ME-Adj-Hier-A	3	1	4	1	1	-
ME-Adj-Hier-C	1	1	2	②	1	-
ME-Adj-Hier-S	1	1	1	1	1	-
ME-Adj-K-means-R	1	1	1	2	1	-
ME-Adj-K-means-A	1	②	1	②	1	-
ME-Adj-K-means-C	1	②	2	②	1	-
ME-Adj-K-means-S	1	1	1	1	1	-
ME-F-Hier-A	4	1	3	1	1	-
ME-F-Hier-C	3	1	2	②	1	-
ME-F-Hier-S	⑦	1	2	❸	-	-
ME-F-K-means-R	1	②	2	②	2	-
ME-F-K-means-A	2	❸	4	②	2	-
ME-F-K-means-C	2	②	2	②	2	-
ME-F-K-means-S	2	1	2	④	④	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 6.9: A summary of the precision results for ME on all algorithms and Benchmark 1 datasets, with use of three external indices. For PBM, the experiments were stopped due to their high computational demand (weeks to complete).

of ME was rather disappointing, with the exception of Leukemia and Lymphoma, across algorithms and external indices.

6.2.6 Consensus

For each of the first five datasets and each clustering algorithm, Consensus is computed for a number of cluster values in the range $[2, 30]$. Following the same experimental setup of Monti et al., for each cluster value k, 500 resampling steps are performed. In each step, 80% of the rows of the matrix are extracted uniformly and at random to create a new dataset, to be given to the clustering algorithm to be clustered in k groups. The prediction of k^* is based on the plot of two curves, $\Delta(k)$ and $\Delta'(k)$, as a function of the number k of clusters. Both curves are defined in Chapter 3. As suggested by Monti et al., the first curve is suitable for hierarchical algorithms while the second suits non-hierarchical ones. The experiment for PBM were aborted since Consensus was very slow (execution on each algorithm was terminated after a week). Contrary to Monti et al. indication, the $\Delta(k)$ curve is computed for all algorithms on the first five datasets, for reasons that will be self-evident shortly. The corresponding plots are available at the following supplementary material web site [3] (Figures section) as Figs. S125-S134. Moreover, the $\Delta'(k)$ curve is also computed for the K-means algorithms, on the same datasets. Recall from Chapter 3 the recommendation to use the Δ' curve instead of the Δ curve for non-hierarchical algorithms as suggested by

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
`ME-FM`-Hier-A	2.8×10^5	2.3×10^5	7.6×10^5	6.5×10^5
`ME-FM`-Hier-C	2.9×10^5	2.3×10^5	7.6×10^5	6.5×10^5
`ME-FM`-Hier-S	2.9×10^5	2.3×10^5	7.7×10^5	6.5×10^5
`ME-FM`-K-means-R	3.6×10^5	3.9×10^5	1.3×10^6	1.1×10^6
`ME-FM`-K-means-A	4.6×10^5	3.6×10^5	1.1×10^6	9.8×10^5
`ME-FM`-K-means-C	4.4×10^5	3.7×10^5	1.1×10^6	9.8×10^5
`ME-FM`-K-means-S	5.3×10^5	3.7×10^5	1.2×10^6	1.0×10^6
`ME-Adj`-Hier-A	2.7×10^5	2.3×10^5	7.6×10^5	6.4×10^5
`ME-Adj`-Hier-C	2.7×10^5	2.3×10^5	7.6×10^5	6.5×10^5
`ME-Adj`-Hier-S	2.7×10^5	2.3×10^5	7.5×10^5	6.5×10^5
`ME-Adj`-K-means-R	3.4×10^5	3.8×10^5	1.3×10^6	1.1×10^6
`ME-Adj`-K-means-A	4.4×10^5	3.5×10^5	1.1×10^6	9.8×10^5
`ME-Adj`-K-means-C	4.2×10^5	4.2×10^5	1.1×10^6	9.9×10^5
`ME-Adj`-K-means-S	5.1×10^5	3.7×10^5	1.2×10^6	1.0×10^6
`ME-F`-Hier-A	2.8×10^5	2.1×10^5	7.3×10^5	6.4×10^5
`ME-F`-Hier-C	2.8×10^5	2.2×10^5	7.4×10^5	6.4×10^5
`ME-F`-Hier-S	2.8×10^5	2.1×10^5	7.4×10^5	6.4×10^5
`ME-F`-K-means-R	3.6×10^5	3.8×10^5	1.3×10^6	1.0×10^6
`ME-F`-K-means-A	4.5×10^5	3.5×10^5	1.1×10^6	9.5×10^5
`ME-F`-K-means-C	4.3×10^5	3.5×10^5	1.1×10^6	9.6×10^5
`ME-F`-K-means-S	5.2×10^5	3.5×10^5	1.1×10^6	1.0×10^6

Table 6.10: A summary of the timing results for `ME` on all algorithms, with use of three external indices. For PBM, the experiments were stopped due to their high computational demand (weeks to complete).

	Precision				
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
Hier-A	⑦	❸	❽	❸	❺
Hier-C	❻	④	❽	5	⑥
Hier-S	2	8	10	❸	10
K-means-R	❻	④	⑦	❸	⑥
K-means-A	⑦	❸	❽	❸	⑥
K-means-C	❻	❸	❽	④	⑥
K-means-S	⑦	④	10	②	⑥
Gold solution	**6**	**3**	**8**	**3**	**5**

Table 6.11: A summary of the precision results for `Consensus` on all algorithms and the first five datasets in `Benchmark 1`. For PBM, the experiments were terminated due to their high computational demand and the corresponding column has been removed from the table.

Monti et al.. Briefly, the reason is the following: $A(k)$ is a value that is expected to behaves like a non-decreasing function of k, for hierarchical algorithms. Therefore, $\Delta(k)$ would be expected to be positive or, when negative, not too far from zero. Such a monotonicity of $A(k)$ is not expected for non-hierarchical algorithms. Therefore, another definition of Δ is needed to ensure a behavior of this function analogous to the hierarchical algorithms. However, from the experiments reported in Table 6.11, for the K-means algorithms, $A(k)$ displays nearly the same monotonicity properties of the hierarchical algorithms. The end result is that Δ can be used for both types of algorithms. Consequently, since the Δ' curves are nearly identical to the $\Delta(k)$ ones, they are omitted. In order to predict the number of clusters in the datasets, for all curves, the rule reported and explained in the Section 3.3.1 is used: take as k^* the abscissa corresponding to the smallest non-negative value where the curve starts to stabilize; that is, no big variation in the curve takes place from that point on. An analysis on the $\Delta(k)$ curves is performed and the precision results are summarized in Table 6.11 and the corresponding timing results in Table 6.12.

As for the precision of `Consensus`, all algorithms perform well, except for Hier-S.

In conclusion, `Consensus` seems to be limited by time demand that makes it not applicable to large datasets. However, on small and medium sized datasets, it is remarkably precise across algorithms. In fact, except for Hier-S, the performance of `Consensus` is among the best and reported in Tables 6.15 and 6.16, for comparison with the performance of the other measures.

6.2.7 FOM

For each algorithm, and each dataset, the same methodology outlined for `WCSS` is followed. The relevant plots are in Figs. S135-S136 at the following supplementary material web site [3] (Figures section). The values resulting from the application of this methodology to the relevant plots are reported in Table 6.13 and 6.14 together with timing results for the relevant datasets. From those results, it is possible to see as `FOM` is algorithm-dependent and gives no useful indication on large datasets. The best performing settings are reported in Tables 6.15, and 6.16 for comparison with the performance of the other measures.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
Hier-A	9.2×10^5	7.9×10^5	2.0×10^6	1.9×10^6
Hier-C	8.7×10^5	6.9×10^5	2.0×10^6	2.0×10^6
Hier-S	9.4×10^5	8.0×10^5	2.0×10^6	1.7×10^6
K-means-R	1.0×10^6	1.3×10^6	3.4×10^6	3.0×10^6
K-means-A	1.3×10^6	1.6×10^6	3.0×10^6	2.6×10^6
K-means-C	1.3×10^6	1.8×10^6	2.9×10^6	2.6×10^6
K-means-S	1.5×10^6	1.8×10^6	3.2×10^6	2.8×10^6

Table 6.12: A summary of the timing results for `Consensus` on all algorithms and the first five datasets in `Benchmark 1`. For PBM, the experiments were terminated due to their high computational demand and the corresponding column has been removed from the table.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	⑦	6	⑥	-
Hier-C	10	④	⑦	7	❺	-
Hier-S	3	7	⑦	9	-	-
K-means-R	⑦	❸	6	9	④	-
K-means-A	⑦	❸	6	6	④	-
K-means-C	⑦	8	❽	④	④	-
K-means-S	❻	❸	❽	8	④	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 6.13: A summary of the precision results for `FOM` on all algorithms and on `Benchmark 1` datasets. Cells with a dash indicate that `FOM` did not give any useful indication.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
Hier-C	1.6×10^3	7.7×10^3	4.5×10^4	1.8×10^4
Hier-S	1.6×10^3	7.4×10^3	4.9×10^5	1.7×10^4
K-means-R	2.9×10^4	1.9×10^5	1.3×10^6	6.7×10^5
K-means-A	2.2×10^4	9.3×10^4	5.5×10^5	2.7×10^5
K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5

Table 6.14: A summary of the timing results for `FOM` on all algorithms and on `Benchmark 1` datasets.

6.3 Relative Merits of Each Measure

The discussion here refers to Tables 6.15 and 6.16. It is evident that the K-means algorithms have superior performance with respect to the hierarchical ones, although Hier-A has an impressive and unmatched performance with `Consensus`.

However, `Consensus` and `FOM` stand out as being the most stable across algorithms. In particular, `Consensus` has a remarkable stability performance across algorithms and datasets.

For large datasets such as PBM, the experiments show that all the measures are severely limited due either to speed (`Clest`, `Consensus`, `Gap-Pc`) or to precision as well as speed (the others). Therefore, this fact stresses even more the need for good data filtering and dimensionality reduction techniques since they may help reduce such datasets to sizes manageable by the measures studied in this chapter.

It is also obvious that, when one takes computer time into account, there is a hierarchy of measures, with `WCSS` being the fastest and `Consensus` the slowest. It is worth pointing out that from Table 6.16 that there is a natural division of methods in two groups: slow (`Clest`, `Consensus`, `Gap`) and fast (the other measures). Since there are at least two orders of magnitude of difference in time performance between the two groups, it seems reasonable to use one of the fast methods to limit the search interval for k^*. One can then use `Consensus` in the narrowed interval. Although it may seem paradoxical, despite its precision performance, `FOM` does not seem to be competitive in this scenario. Indeed, it is only marginally better than the best performing setting of `WCSS` but at least an order of magnitude slower in time.

When one does not account for time, `Consensus` seems to be the clear winner since it offers good precision performance across algorithms at virtually the same price in terms of time performance.

It is also important pointing out that the three instances of the Stability Measure paradigm have quite diverging performances. Such a fact gives evidence that care must be exercised in taking full advantage of such a powerful paradigm. Indeed, only `Consensus` seems to take full advantage of the repeated data generation. A possible reason for this is in the different implementation of the Stability Statistic paradigm to collect the statistic. In fact, `Consensus` builds a matrix, the consensus matrix, that contains very punctual and global information about the cluster structure of the dataset, while the other two measures try to infer that structure by first splitting the dataset in two subsets and then by using a synoptic function (an external index) to assess the similarity between the partitions. That is, those latter two measures use a coarse assessment of consistency. Moreover, `ME` uses the same algorithm for both of the datasets generated. Probably that induces a big dependency of the measure on the clustering algorithm.

Considering the results of Tables 6.15 and 6.16, a promising avenue of research is to design fast approximation algorithms for the computation of the slowest measures, in particular `Consensus`. Finally, it is worth pointing out that `Gap`, `Clest`, `ME` and `Consensus` have various parameters that a user needs to specify. Those choices may affect both time performance and precision. However, no parameter tuning is available in the Literature.

The next chapter addresses those issues. Indeed, a first study of the best parameter setting for `Consensus` is provided. Moreover, an approximation of several measures is presented, with particular focus on `Gap` and `Consensus`. Moreover, a general scheme speeding up the Stability Statistic paradigm is also provided.

	Precision				
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
`WCSS`-K-means-C	⑤	❸	❽	8	④
`FOM`-Hier-A	⑦	❸	⑦	6	⑥
`FOM`-K-means-C	⑦	8	❽	④	④
`FOM`-K-means-S	❻	❸	❽	8	④
`Clest-F`-K-means-R	❻	❸	15	②	④
`Clest-FM`-K-means-R	8	④	❽	②	④
`Consensus`-Hier-A	⑦	❸	❽	❸	❺
`Consensus`-Hier-C	❻	④	❽	5	⑥
`Consensus`-K-means-R	❻	④	⑦	❸	⑥
`Consensus`-K-means-A	⑦	❸	❽	❸	⑥
`Consensus`-K-means-C	❻	❸	❽	④	⑥
`Consensus`-K-means-S	⑦	④	10	②	⑥
Gold solution	**6**	**3**	**8**	**3**	**5**

Table 6.15: A summary of the precision for the best performances obtained by each measure. The PBM dataset has been excluded because no measure gave useful information about its cluster structure.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
`WCSS`-K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
`FOM`-Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
`FOM`-K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
`FOM`-K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5
`Clest-F`-K-means-R	1.2×10^6	-	-	-
`Clest-FM`-K-means-R	1.2×10^6	-	-	-
`Consensus`-Hier-A	9.2×10^5	7.9×10^5	2.0×10^6	1.9×10^6
`Consensus`-Hier-C	8.7×10^5	6.9×10^5	2.0×10^6	2.0×10^6
`Consensus`-K-means-R	1.0×10^6	1.3×10^6	3.4×10^6	3.0×10^6
`Consensus`-K-means-A	1.3×10^6	1.6×10^6	3.0×10^6	2.6×10^6
`Consensus`-K-means-C	1.3×10^6	1.8×10^6	2.9×10^6	2.6×10^6
`Consensus`-K-means-S	1.5×10^6	1.8×10^6	3.2×10^6	2.8×10^6

Table 6.16: A summary of the timing for the best performances obtained by each measure. The PBM dataset has been excluded because no measure gave useful information about its cluster structure.

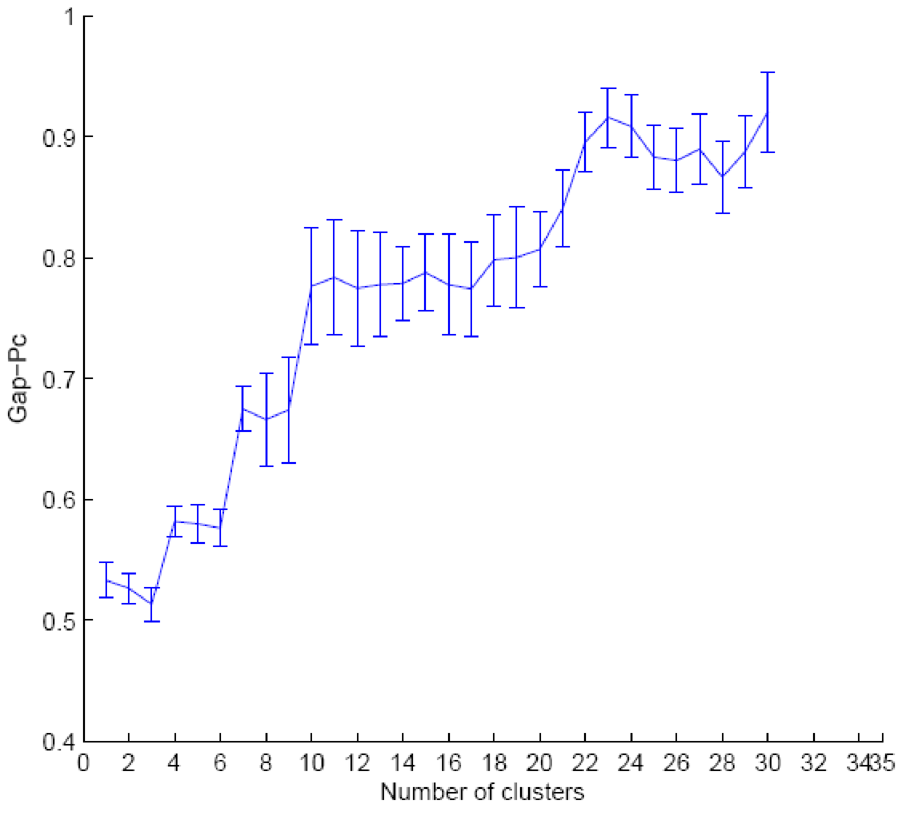

Figure 6.1: The `Gap-Pc` curve on the Leukemia dataset, with use of the Hier-S algorithm. At each point, error bars indicate the variation of the curve across simulations. The curve shows a first maximum at $k = 1$, yielding a prediction of $k^* = 1$, the next maximum is at $k = 4$, which is very close to the number of classes $k^* = 3$.

Chapter 7

Speedups of Internal Validation Measures Based on Approximations

One open question that was made explicit by the study of the previous chapter is the design of a data-driven internal validation measure that is both precise and fast, and capable of granting scalability with dataset size. Such a lack of scalability for the most precise internal validation measures is one of the main computational bottlenecks in the process of cluster evaluation for microarray data analysis. Its elimination is far from trivial [99] and even partial progress on this problem is perceived as important. In the research area embodying the design and analysis of algorithms, when a problem is computationally difficult, a usual approach to its solution is to design fast heuristics and/or provably good approximation algorithms, in order to obtain solution that are "close" to the ones that would be produced by the exact algorithms.

In this chapter, the algorithmic approach just outlined is investigated in a systematic way in the realm of internal validation measures, with the goal of narrowing the time performance gap, identified in the previous chapter, between the most precise and the fastest measures. In particular, several algorithmic approximations and two general approximations schemes are presented.

7.1 An Approximation of WCSS

The approximation of `WCSS` proposed here is based on the idea of reducing the number of executions of a clustering algorithm C for the computation of $\texttt{WCSS}(k)$, for each k in a given interval $[1, k_{max}]$. In fact, given an integer $R > 0$, which is referred to as refresh step, the approximate algorithm to compute `WCSS` uses algorithm C to obtain a clustering solution with k clusters, only for values of k multiples of R. For all other k's, a clustering solution is obtained by merging two clusters in a chosen clustering solution already available. The procedure in Fig. 7.1 gives the high level details. It takes as input R, C, D and k_{max}. Algorithm C must be able to take as input a clustering solution with k clusters and refine it to give as output a clustering solution with the same number of clusters.

Technically, the main idea in the approximation scheme is to interleave the execution of a partitional clustering algorithm C with a merge step typical of agglomerative clustering. The gain in speed is realized by having a fast merge step, based on $k+1$ clusters, to obtain k clusters instead of a new full fledged computation, starting from scratch, of the algorithm C to obtain the same number of clusters. The approximation scheme would work also for hierarchical algorithms, provided that they comply with the requirement that, given as input a dataset, they will return a partition into k groups. However, in this circumstance, the approximation scheme would be a nearly exact replica of the hierarchical algorithm. In conclusion, a general approximation scheme is proposed, where the gain is realized when the

```
WCSS-R($R, C, D, k_{max}$)

1. Let $P_{k_{max}}$ be the partition of $D$ into $k_{max}$ cluster with the use of $C$
2. Compute WCSS($k_{max}$) using $P_{k_{max}}$
 for $k \leftarrow k_{max} - 1$ downto 1 do
   begin
     comment: Merge
3.   Merge the two clusters in $P_{k+1}$ with minimum Euclidean distance
     between centroids to obtain a temporary clustering solution
     $P'_k$ with $k$ clusters
     comment: Refresh
4.   if ($R = 0$) or ($k$ mod $R > 0$)
  then $P_k \leftarrow P'_k$
  else Compute new $P_k$ based on $P'_k$. That is, $P'_k$ is given as input to $C$, as an
    initial partition of $D$ in $k$ clusters, and $P_k$ is the result of that computation.
5.   Compute WCSS($k$) using $P_k$
   end
```

Figure 7.1: The WCSS-R procedure

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	10	❸	3	6	❺	-
Hier-C	10	❸	⑦	8	9	-
Hier-S	8	10	⑦	9	-	-
R-R0	⑤	④	⑨	❸	④	-
R-R2	⑦	5	15	④	④	-
R-R5	❻	5	⑨	5	④	-
K-means-R	4	❸	3	8	④	-
K-means-A	4	❸	⑦	6	❺	-
K-means-C	⑤	❸	❽	8	④	-
K-means-S	3	④	⑦	8	24	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.1: A summary of the precision results for `WCSS` of Table 6.1 with the addition of its approximation. Cells with a dash indicate that `WCSS` did not give any useful indication.

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
Hier-A	1.1×10^3	4.0×10^2	2.1×10^3	1.9×10^3
Hier-C	7.0×10^2	4.0×10^2	1.7×10^3	1.4×10^3
Hier-S	2.6×10^3	6.0×10^2	3.2×10^3	3.8×10^3
R-R0	1.2×10^3	8.0×10^2	4.1×10^3	3.0×10^3
R-R2	1.3×10^3	8.0×10^2	5.3×10^3	3.2×10^3
R-R5	1.2×10^3	8.0×10^2	4.6×10^3	3.2×10^3
K-means-R	2.4×10^3	2.0×10^3	8.4×10^3	8.4×10^3
K-means-A	2.3×10^3	1.3×10^3	5.4×10^3	5.8×10^3
K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
K-means-S	2.6×10^3	1.6×10^3	7.3×10^3	7.4×10^3

Table 7.2: A summary of the timing results for `WCSS` of Table 6.2 with the addition of its approximation.

merge step is faster than a complete computation of a clustering algorithm C. In this thesis, experiments have been conducted with K-means-R on `Benchmark 1` datasets, and with values of the refresh step $R = 0, 2, 5$, i.e., the partitional clustering algorithm is used only once, every two and five steps, respectively. The corresponding results are summarized in Tables 7.1 and 7.2, together with the results of `WCSS` already reported in Tables 6.1 and 6.2. As is self-evident from the results in the former tables, the approximation has a better predicting power than the original `WCSS` curve (obtained via all other clustering algorithms one has experimented with). In fact, the approximation is among the best performers. Moreover, depending on the dataset, it is from a few times to an order of magnitude faster than the K-means algorithms (see Table 7.2).

7.2 A Geometric Approximation of Gap Statistics: G-Gap

The geometric interpretation of Gap given in Chapter 2 and the behavior of the WCSS curve across null models suggests a fast approximation, which it is referred to as G-Gap. The intuition, based on experimental observations, is that one can skip the entire simulation phase of Gap, without compromising too much the accuracy of the prediction of k^*. Indeed, based on the WCSS curve, the plot of the log WCSS curve one expects, for a given clustering algorithm and null model, is a straight line with a slope somewhat analogous to that of the log WCSS curve and dominating it (see Fig. 2.3). Therefore, one can simply identify the "knee" in the WCSS by translating the end-points of the log WCSS curve on the original dataset by a given amount a, to obtain two points g_s and g_e. Those two points are then joined by a straight line, which is used to replace the null model curve to compute the segment lengths used to predict k^*, i.e, the first maximum among those segment lengths as k increases. An example is provided in Fig. 7.2 with the WCSS curve of Fig. 2.1. The prediction is $k^* = 3$, which is very close to the correct $k^* = 2$. It is worth pointing out that the use of the WCSS curve in the figure is to make clear the behavior of the segment lengths, which would be unnoticeable with the log WCSS curve, although the result would be the same.

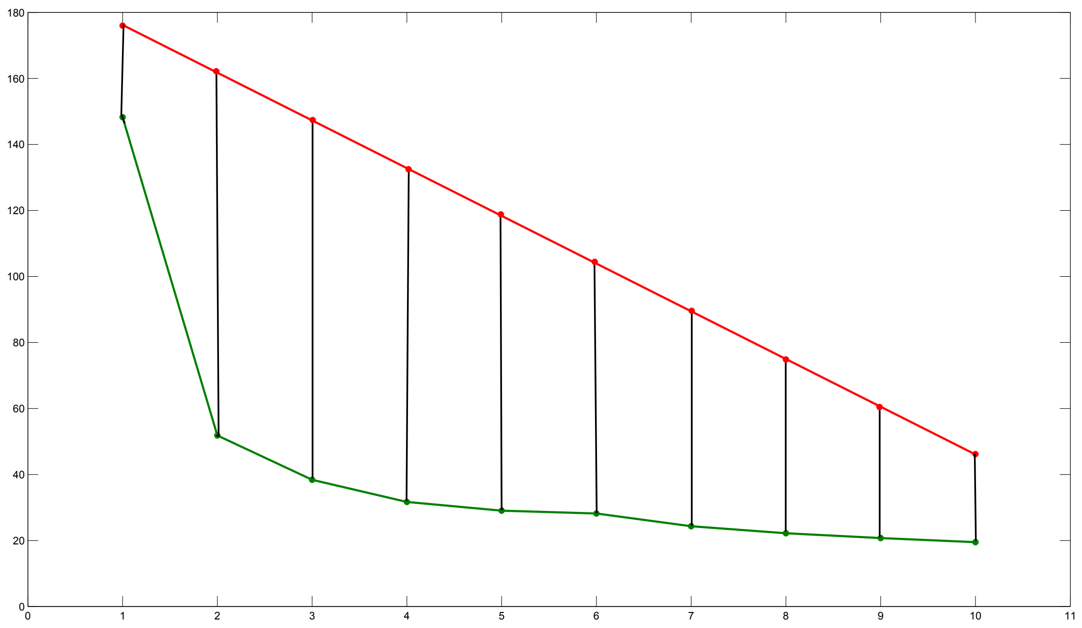

Figure 7.2: The G-Gap Heuristic. The curve in green is a WCSS curve obtained on the dataset of Fig. 1.4(a) with the use of the K-means algorithm. The line in red is obtained by projecting upward the end points of the WCSS curve by a units and then joining them. It is a heuristic approximation of WCSS for a null model. The vertical lines have the same role as in Gap and the rule to identify k^* is the same, yielding a value $k^* = 3$, a value very close to the correct number of classes (two) in the dataset.

As for G-Gap, the geometric approximation of Gap, each algorithm and each dataset, the corresponding WCSS curve and its approximations has been computed in the interval $[1, 30]$ on Benchmark 1 datasets. The rule described above has been applied to get the value of k^*.

The corresponding results are summarized in Tables 7.3 and 7.4 with the addition of the results of Gap reported in Tables 6.5 and 6.6. As it is evident from Tables 7.3, the overall performance of G-Gap is clearly superior to Gap, irrespective of the null model. Moreover, depending on the dataset, it is from two to three orders of magnitude faster (see Table 7.4).

7.3 An Approximation of FOM

Recalling from Chapter 2 that both WCSS and FOM use the same criteria that is used in order to infer k^*. Such an analogy between FOM and WCSS immediately suggest to extend some of the knowledge available about WCSS to FOM, as follows:

- The approximation of FOM is based on exactly the same ideas and schemes presented for the approximation of WCSS. Indeed, FOM(e, k) in equation (2.20) can be approximated in exactly the same way as WCSS(k). Then, one uses equation (2.21) to approximate FOM. We denote those approximations as FOM-R.
- The G-Gap idea can be extended verbatim to FOM to make it automatic and to obtain G-FOM.
- The KL technique can be extended to FOM, although the extension is subtle. Indeed, a verbatim extension of it would yield poor results (experiments not shown). Rather, consider formula (2.17), with WCSS(k) substituted by FOM(k). As k increases towards k^*, $DIFF(k)$ increases to decrease sharply and then assume nearly constant values as it moves away from k^*. Fig. 7.3 provides a small example of this behavior. So, one can take as k^* the abscissa corresponding to the maximum of $DIFF(k)$ in the interval $[3, k_{max}]$. This method is referred to as DIFF-FOM.

For each algorithm, each of the FOM approximations (denoted FOM-R-R0, FOM-R-R2, FOM-R-R5, respectively) and each dataset in Benchmark 1, the same methodology outlined for WCSS and its approximation has been followed. The relevant plots are in Figs. S135-S136 at the following supplementary material web site [3] (Figures section). The values resulting from the application of this methodology to the relevant plots are reported in Table 7.5, while the timing results for the relevant datasets are reported in Table 7.6 with the addition of the results for FOM reported in Tables 6.13 and 6.14, respectively. Using the same experimental setting, G-FOM and DIFF-FOM, the extensions of FOM proposed here, are computed in order to predict k^*. The results are in Tables 7.7-7.8 and 7.9-7.10, respectively. As those results show, G-FOM does not perform as well as the other two. Moreover, both FOM and DIFF-FOM are algorithm-dependent and give no useful indication on large datasets. As for the approximations of FOM, i.e., FOM-R-R0, FOM-R-R2, FOM-R-R5, they compare very well with the K-means algorithms in terms of precision and they are an order of magnitude faster.

7.4 An Exhaustive Study of the Consensus Parameters

It is helpful for the discussion to highlight, here, some key facts about Consensus, summarizing the detailed description of the procedure presented in Chapter 3. For a given number of clusters, Consensus computes a certain number of clustering solutions (resampling step), each from a sample of the original data (subsampling). The performance of Consensus depends on two parameters: the number of resampling steps H and the percentage of subsampling p, where p states how large the sample must be. From each clustering

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
`G-Gap`-Hier-A	⑦	❸	1	②	3	1
`G-Gap`-Hier-C	⑦	❸	2	②	7	4
`G-Gap`-Hier-S	⑤	❸	1	1	1	5
`G-Gap`-R-R0	2	7	2	④	❺	4
`G-Gap`-R-R2	3	②	2	②	❺	6
`G-Gap`-R-R5	⑤	④	2	②	④	6
`G-Gap`-K-means-R	⑦	❸	4	④	⑥	5
`G-Gap`-K-means-A	4	❸	1	②	⑥	4
`G-Gap`-K-means-C	⑤	❸	2	8	⑥	5
`G-Gap`-K-means-S	3	❸	1	1	1	1
`Gap-Ps`-Hier-A	1	④	1	6	3	1
`Gap-Ps`-Hier-C	1 or 2	④	2	1 or 25	7	15
`Gap-Ps`-Hier-S	1	1	1	1	1	1
`Gap-Ps`-K-means-R	❻ or ⑦	④ or 5	3	8	9	7
`Gap-Ps`-K-means-A	⑦	❸	1	8	7	9
`Gap-Ps`-K-means-C	⑦	④	2	1 or 25	12	6
`Gap-Ps`-K-means-S	9	❸	1	1	7	8
`Gap-Pc`-Hier-A	1	❸ or ④	1	1	1 or 2 or 3	-
`Gap-Pc`-Hier-C	1	④	1	1	3	-
`Gap-Pc`-Hier-S	1	1	1	1	1	-
`Gap-Pc`-K-means-R	2	1	1	1	④	-
`Gap-Pc`-K-means-A	2	④	1	1	3	-
`Gap-Pc`-K-means-C	2	1	1	1	④	-
`Gap-Pc`-K-means-S	3	1	1	1	1	-
`Gap-Pr`-Hier-A	3	④	1	6	3	1
`Gap-Pr`-Hier-C	⑦	④	1	1 or 25	16	1
`Gap-Pr`-Hier-S	1 or ❻	1	2	1	1	2
`Gap-Pr`-K-means-R	❻	④	5	8	8	8
`Gap-Pr`-K-means-A	8	④	1	8	13	4
`Gap-Pr`-K-means-C	⑤	6	1	1 or 25	8	1
`Gap-Pr`-K-means-S	⑦	❸	2	1	11	1
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.3: A summary of the precision results for `Gap` of Table 6.5 with the addition of its approximations. For `Gap-Pc`, on PBM, the experiments were stopped due to their high computational demand.

	Timing			
	CNS Rat	Leukemia	NCI60	Limphoma
G-Gap-Hier-A	1.1×10^3	4.0×10^2	2.0×10^3	1.9×10^3
G-Gap-Hier-C	7.0×10^2	4.0×10^2	1.7×10^3	1.4×10^3
G-Gap-Hier-S	2.6×10^3	6.0×10^2	3.2×10^3	3.8×10^3
G-Gap-R-R0	1.2×10^3	8.0×10^2	4.0×10^3	3.0×10^3
G-Gap-R-R2	1.3×10^3	8.0×10^2	5.2×10^3	3.2×10^3
G-Gap-R-R5	1.2×10^3	8.0×10^2	4.5×10^3	3.2×10^3
G-Gap-K-means-R	2.4×10^3	2.0×10^3	8.3×10^3	8.4×10^3
G-Gap-K-means-A	2.3×10^3	1.3×10^3	5.3×10^3	5.8×10^3
G-Gap-K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
G-Gap-K-means-S	2.6×10^3	1.6×10^3	7.3×10^3	7.4×10^3
Gap-Ps-Hier-A	2.7×10^5	1.4×10^5	6.1×10^5	6.4×10^5
Gap-Ps-Hier-C	2.3×10^5	1.1×10^4	3.4×10^5	3.2×10^5
Gap-Ps-Hier-S	6.1×10^5	1.9×10^5	1.1×10^6	1.4×10^6
Gap-Ps-K-means-R	8.4×10^5	5.0×10^5	1.1×10^6	1.0×10^6
Gap-Ps-K-means-A	6.1×10^5	4.7×10^5	1.1×10^6	1.0×10^6
Gap-Ps-K-means-C	6.0×10^5	6.1×10^5	8.8×10^5	7.6×10^5
Gap-Ps-K-means-S	9.1×10^5	6.5×10^5	2.1×10^6	1.8×10^6
Gap-Pc-Hier-A	3.2×10^5	3.7×10^5	8.1×10^5	6.1×10^5
Gap-Pc-Hier-C	1.9×10^5	1.9×10^5	7.1×10^5	5.8×10^5
Gap-Pc-Hier-S	7.7×10^5	3.1×10^5	1.4×10^6	1.3×10^6
Gap-Pc-K-means-R	4.9×10^5	8.0×10^5	1.8×10^6	1.8×10^6
Gap-Pc-K-means-A	3.8×10^5	7.0×10^5	1.3×10^6	1.4×10^6
Gap-Pc-K-means-C	4.1×10^5	5.8×10^5	1.3×10^6	1.2×10^6
Gap-Pc-K-means-S	6.5×10^5	7.6×10^5	2.4×10^6	2.0×10^6
Gap-Pr-Hier-A	2.5×10^5	1.6×10^5	3.3×10^5	3.8×10^5
Gap-Pr-Hier-C	1.3×10^5	1.4×10^5	3.2×10^5	3.7×10^5
Gap-Pr-Hier-S	6.8×10^5	1.9×10^5	1.1×10^6	1.4×10^6
Gap-Pr-K-means-R	8.6×10^5	5.4×10^5	1.5×10^6	9.4×10^5
Gap-Pr-K-means-A	7.4×10^5	5.0×10^5	8.7×10^5	1.0×10^6
Gap-Pr-K-means-C	6.7×10^5	4.6×10^5	8.6×10^5	1.0×10^6
Gap-Pr-K-means-S	1.2×10^6	5.4×10^5	1.8×10^6	2.3×10^6

Table 7.4: A summary of the timing results for Gap of Table 6.6 with the addition of its approximations. For Gap-Pc, on PBM, the experiments were stopped due to their high computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	⑦	6	⑥	-
Hier-C	10	④	⑦	7	❺	-
Hier-S	3	7	⑦	9	-	-
R-R0	10	5	⑦	④	7	-
R-R2	8	5	❽	5	❺	-
R-R5	❻	❸	⑦	5	❺	-
K-means-R	⑦	❸	6	9	④	-
K-means-A	⑦	❸	6	6	④	-
K-means-C	⑦	8	❽	④	④	-
K-means-S	❻	❸	❽	8	④	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.5: A summary of the precision results for `FOM` of Table 6.13 with the addition of its approximations. Cells with a dash indicate that `FOM` did not give any useful indication.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
Hier-C	1.6×10^3	7.7×10^3	4.5×10^4	1.8×10^4
Hier-S	1.6×10^3	7.4×10^3	4.9×10^5	1.7×10^4
R-R0	2.6×10^3	3.1×10^4	1.7×10^5	5.3×10^4
R-R2	3.4×10^3	3.8×10^4	2.2×10^5	7.2×10^4
R-R5	3.9×10^3	3.7×10^4	2.1×10^5	7.6×10^4
K-means-R	2.9×10^4	1.9×10^5	1.3×10^6	6.7×10^5
K-means-A	2.2×10^4	9.3×10^4	5.5×10^5	2.7×10^5
K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5

Table 7.6: A summary of the timing results for `FOM` of Table 6.14 with the addition of its approximations.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	3	❸	⑦	②	8	2
Hier-C	10	④	2	④	8	2
Hier-S	⑦	②	2	②	2	2
R-R0	2	7	2	5	7	4
R-R2	⑦	5	2	5	8	4
R-R5	4	④	2	6	⑥	4
K-means-R	⑦	5	6	8	⑥	7
K-means-A	2	❸	⑦	②	⑥	6
K-means-C	2	④	2	④	7	6
K-means-S	3	5	2	②	⑥	8
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.7: A summary of the results for `G-FOM` on all algorithms and on all datasets. The columns under the label precision indicate the number of clusters predicted by `G-FOM`, while the remaining four indicate the timing in milliseconds for the execution of the corresponding experiment. Cells with a dash indicate that `G-FOM` did not give any useful indication.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
Hier-C	1.6×10^3	7.7×10^3	4.5×10^4	1.8×10^4
Hier-S	1.6×10^3	7.4×10^3	4.9×10^5	1.7×10^4
R-R0	2.6×10^3	3.1×10^4	1.7×10^5	5.3×10^4
R-R2	3.4×10^3	3.8×10^4	2.2×10^5	7.2×10^4
R-R5	3.9×10^3	3.7×10^4	2.1×10^5	7.6×10^4
K-means-R	2.9×10^4	1.9×10^5	1.3×10^6	6.7×10^5
K-means-A	2.2×10^4	9.3×10^4	5.5×10^5	2.7×10^5
K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5

Table 7.8: A summary of the precision results for `G-FOM` on all algorithms and on all datasets. Cells with a dash indicate that `G-FOM` did not give any useful indication.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
K-means-R	4	❸	4	❸	3	4
K-means-A	⑦	❸	3	6	3	8
K-means-C	⑦	❸	⑦	④	3	5
K-means-S	⑦	❸	12	8	3	10
R-R0	10	④	17	④	3	3
R-R5	4	❸	11	❸	3	4
R-R2	⑦	❸	17	❸	3	7
Hier-A	⑦	❸	3	6	3	25
Hier-C	9	❸	⑦	7	3	7
Hier-S	20	7	22	9	7	20
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.9: A summary of the precision results for `DIFF-FOM` on all algorithms and on all datasets. Cells with a dash indicate that `DIFF-FOM` did not give any useful indication.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
Hier-C	1.6×10^3	7.7×10^3	4.5×10^4	1.8×10^4
Hier-S	1.6×10^3	7.4×10^3	4.9×10^5	1.7×10^4
R-R0	2.6×10^3	3.1×10^4	1.7×10^5	5.3×10^4
R-R2	3.4×10^3	3.8×10^4	2.2×10^5	7.2×10^4
R-R5	3.9×10^3	3.7×10^4	2.1×10^5	7.6×10^4
K-means-R	2.9×10^4	1.9×10^5	1.3×10^6	6.7×10^5
K-means-A	2.2×10^4	9.3×10^4	5.5×10^5	2.7×10^5
K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5

Table 7.10: A summary of the timing results for `DIFF-FOM` on all algorithms and on all datasets. Cells with a dash indicate that `DIFF-FOM` did not give any useful indication.

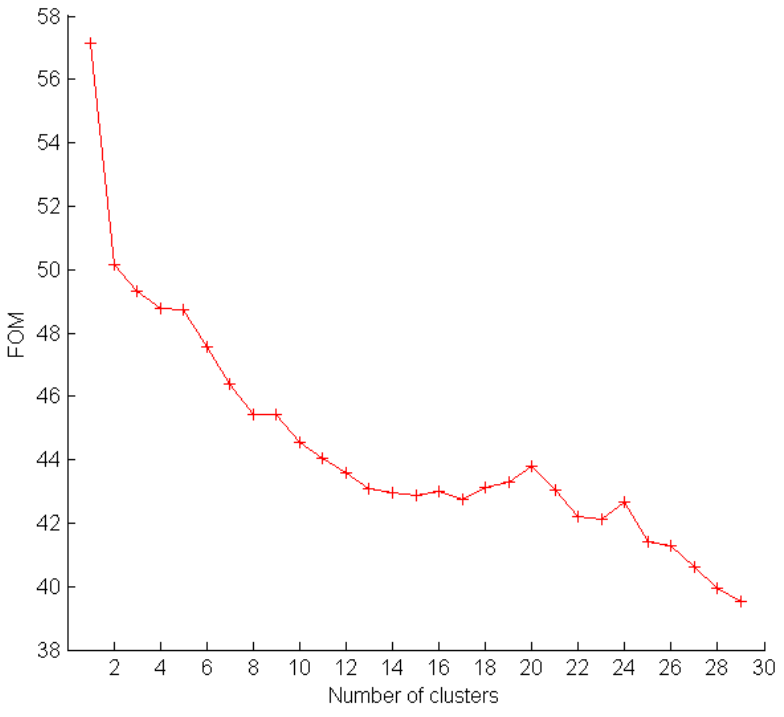

Figure 7.3: The `FOM` curve computed on the Leukemia dataset with K-means-R. As for `WCSS`, the "knee" in the plot indicates the correct number of clusters in the dataset: $k^* = 3$.

solution, a corresponding connectivity matrix is computed: each entry in that matrix indicates whether a pair of elements is in the same cluster or not. For the given number of clusters, the consensus matrix is a normalized sum of the corresponding H connectivity matrices. Intuitively, the consensus matrix indicates the level of agreement of clustering solutions that have been obtained via independent sampling of the dataset.

Monti et al., in their seminal paper, set $H = 500$ and $p = 80\%$, without any experimental or theoretical justification. For this reason and based also on an open problem mentioned in Chapter 6 and in [62], here several experiments with different parameter settings of H and p are performed, in order to find the "best" precision-time trade-off, when `Consensus` is regarded both as an internal validation measure and as a procedure that computes a similarity/distance matrix. In particular, using the hierarchical algorithms and K-means, experiments with $H = 500, 250, 100$ and $p = 80\%, 66\%$ have been performed, respectively, reporting the precision values and times. The choice of the value of p is justified by the results reported in [16, 48]. Intuitively, a value of p smaller then 66% would fail to capture the entire cluster structure present in the data. The results in this section will show, it is worthy to anticipate, that a simple reduction in terms of H and p is not enough to grant a good precision-time trade-off. Such a finding, together with the state of the art outlined in Chapter 6, motivates a strong interest in the design of alternative methods, such as fast heuristics that are discussed in depth in Section 7.5. As for datasets in `Benchmark 2`, only the experiments with $H = 250$ and $p = 80\%$ are performed, reporting precision results for all and timing only for the microarray data, since the timing results for the artificial datasets are redundant. The choice for this parameter setting for `Consensus` is justified when the results of the experiments on the datasets in `Benchmark 1` are discussed. Moreover, the study of a proper parameter setting for `Consensus` is limited only to the `Benchmark 1` datasets for pragmatic reasons: that choice allows to complete the (rather high) required number of experiments in a reasonable amount of time.

Due to its high computational demand (see Chapter 5), experiments only with $H = 250$ and $p = 80\%$ for NMF have been performed. They are reported in the relevant Table together with the results of the other algorithms, but they are discussed separately. The choice for this parameter setting for `Consensus` when used in conjunction with NMF is justified when the results of the experiments are discussed.

7.4.1 Consensus as an Internal Validation Measure

The experiments summarized here refer to the `Benchmark 1` datasets. Separate tables for each experimental setup are reported: they are Tables 7.11-7.22. For each dataset and each clustering algorithm, `Consensus` for a number of cluster values in the range $[2, 30]$ is computed, while, for Leukemia, the range $[2, 25]$ is used when $p = 66\%$, due to the small size of the dataset. The prediction value, k^*, is based on the plot of the $\Delta(k)$ curve (defined in Chapter 3) as indicated in Chapter 6 and in [62]. The corresponding plots are available at the following supplementary material web site [4], in the Figures section, as Figs. S1-S10 for $p = 80\%$ and $H = 500$, and Figs. S11-S24 for $p = 80\%$ and $H = 250$, Figs. S25-S34 for $p = 80\%$ and $H = 100$. For $p = 66\%$ the relevant figures are Figs. S35-S44, S45-S54 and S55-S64 for $H = 500$, $H = 250$ and $H = 100$, respectively.

For $p = 80\%$, the precision results reported in Tables 7.11-7.16 show there is very little difference between the results obtained for $H = 500$ and $H = 250$. That is in contrast with the results for $H = 100$, where many prediction values are very far from the gold solution for the corresponding dataset, e.g., the Lymphoma dataset. Such a finding seems to indicate that, in order to find a consensus matrix which captures well the inherent structure of the dataset, one needs a sensible number of connectivity matrices. The results for a subsampling value of $p = 66\%$ confirms that the number of connectivity matrices one needs to compute is more relevant than the percentage of the data matrix actually used to compute them. Indeed, although it is obvious that a reduction in the number of resampling steps results in a saving in terms of execution time, it is less obvious that for subsampling values $p = 66\%$ and $p = 80\%$, there is no substantial difference in the results, both in terms of precision and of time.

In regard to NMF, only the parameter setting $H = 250$ and $p = 80\%$ for this experiments is used, since it seems to be the most promising (as determined by the use of the other algorithms). The results are reported in Table 7.13. Even so, the inefficiencies of `Consensus` compound with those of NMF; that is, the relatively large number of connectivity matrices needed by `Consensus` and the well-known slow convergence of NMF for the computation of a clustering solution, since connectivity matrices are obtained from clustering solutions. The end-result is a slow-down of one order of magnitude with respect to `Consensus` used in conjunction with other clustering algorithms. As a consequence, NMF and `Consensus` can be used together on a conventional PC only for relatively small datasets. In fact, the experiments for Yeast and PBM, the two largest datasets with which one has experimented, were stopped after four days.

It is also worth to point out that, although the parameter setting $H = 250$ and $p = 80\%$ grants a faster execution of `Consensus` with respect to the original setting by Monti et al., the experiments on the PBM dataset were stopped after four days on all algorithms. That is, the largest of the datasets used here is still "out of reach" of `Consensus` even with a tuning of the parameters aimed at reducing its computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	❽	❸	❺	-
Hier-C	❻	④	❽	5	⑥	-
Hier-S	2	8	10	❸	10	-
K-means-R	❻	④	⑦	❸	⑥	-
K-means-A	⑦	❸	❽	❸	⑥	-
K-means-C	❻	❸	❽	④	⑥	-
K-means-S	⑦	④	10	②	⑥	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.11: A summary of the precision results for `Consensus` with $H = 500$ and $p = 80\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	9.2×10^5	7.9×10^5	2.0×10^6	1.9×10^6	7.7×10^7	-
Hier-C	8.7×10^5	6.9×10^5	2.0×10^6	2.0×10^6	8.2×10^7	-
Hier-S	9.4×10^5	8.0×10^5	2.0×10^6	1.7×10^6	8.2×10^7	-
K-means-R	1.0×10^6	1.3×10^6	3.4×10^6	3.0×10^6	5.5×10^7	-
K-means-A	1.3×10^6	1.6×10^6	3.0×10^6	2.6×10^6	1.1×10^8	-
K-means-C	1.3×10^6	1.8×10^6	2.9×10^6	2.6×10^6	9.3×10^7	-
K-means-S	1.5×10^6	1.8×10^6	3.2×10^6	2.8×10^6	9.7×10^7	-

Table 7.12: A summary of the timing results for `Consensus` with $H = 500$ and $p = 80\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

In conclusion, this experiments show that an effective parameter setting for `Consensus` is $H = 250$ and $p = 80\%$: in fact Table 7.13 displays the best trade-off between precision and time. Moreover, the experiments also show that inefficiencies of the `Consensus` methodology are due to the large number of connectivity matrices that are required to compute a reliable consensus matrix, rather than to the size of the sample taken from the data matrix that is then used to compute them. This is particularly important since a slow clustering algorithm, e.g., NMF, used in conjunction with the methodology makes it worthless on conventional computers.

7.4.2 Consensus and Similarity Matrices

One concentrates on two experiments that, together, assess the ability of `Consensus` to produce a similarity/distance matrix that actually improves the performance of clustering algorithms. In particular, following Monti et al., this thesis concentrates on hierarchical algorithms. As in the previous section, only the `Benchmark 1` datasets are used.

The first experiment is as follows: for each dataset and each hierarchical algorithm considered here, one takes the consensus matrix corresponding to the number of clusters k^* predicted by `Consensus`. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand`

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	❽	❸	❺	-
Hier-C	❻	④	❽	5	⑥	-
Hier-S	2	❸	10	②	10	-
K-means-R	❻	④	⑦	④	⑥	-
K-means-A	⑦	❸	❽	❸	⑥	-
K-means-C	❻	❸	❽	④	⑥	-
K-means-S	⑦	5	⑨	②	⑥	-
NMF-R	❻	④	⑦	④	-	-
NMF-A	⑦	❸	2	❸	-	-
NMF-C	⑤	④	⑦	④	-	-
NMF-S	2	8	⑨	②	-	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.13: A summary of the precision results for `Consensus` with $H = 250$ and $p = 80\%$, on all algorithms and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	8.9×10^5	6.0×10^5	1.4×10^6	1.3×10^6	5.0×10^7	-
Hier-C	8.1×10^5	6.0×10^5	1.3×10^6	1.3×10^6	4.8×10^7	-
Hier-S	4.3×10^5	6.2×10^5	1.0×10^5	1.2×10^6	4.8×10^7	-
K-means-R	5.6×10^5	3.7×10^5	1.2×10^6	1.1×10^6	2.7×10^7	-
K-means-A	1.0×10^6	6.4×10^5	1.8×10^6	1.7×10^6	5.6×10^7	-
K-means-C	9.8×10^5	6.5×10^5	1.7×10^6	1.3×10^6	5.3×10^7	-
K-means-S	1.2×10^6	4.7×10^5	1.2×10^6	1.2×10^6	5.7×10^7	-
NMF-R	1.1×10^8	1.3×10^7	6.4×10^7	7.7×10^7	-	-
NMF-A	3.0×10^7	3.0×10^7	1.3×10^7	1.6×10^7	-	-
NMF-C	3.0×10^7	4.4×10^6	1.3×10^7	1.7×10^7	-	-
NMF-S	3.6×10^7	4.7×10^6	1.3×10^7	1.6×10^7	-	-

Table 7.14: A summary of the timing results for `Consensus` with $H = 250$ and $p = 80\%$, on all algorithms and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	❻	❸	⑦	❸	⑥	-
Hier-C	⑦	④	❽	6	❺	-
Hier-S	2	9	⑨	10	2	-
K-means-R	❻	❸ - ④	⑦	6	⑥	-
K-means-A	⑦	❸	⑦	6	⑥	-
K-means-C	⑤	④	❽	6	⑥	-
K-means-S	8	8	⑨	8	⑥	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.15: A summary of the precision results for `Consensus` with $H = 100$ and $p = 80\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	3.5×10^5	2.6×10^5	4.2×10^5	3.8×10^5	2.9×10^7	-
Hier-C	3.3×10^5	2.7×10^5	3.9×10^5	3.9×10^5	2.9×10^7	-
Hier-S	3.5×10^5	2.8×10^5	3.9×10^5	4.2×10^5	2.8×10^7	-
K-means-R	3.6×10^5	3.4×10^5	7.4×10^5	6.2×10^5	2.3×10^7	-
K-means-A	4.2×10^5	1.9×10^5	7.2×10^5	5.2×10^5	7.5×10^6	-
K-means-C	4.4×10^5	2.6×10^5	6.7×10^5	5.5×10^5	3.3×10^7	-
K-means-S	4.8×10^5	2.8×10^5	6.2×10^5	5.8×10^5	3.7×10^7	-

Table 7.16: A summary of the timing results for `Consensus` with $H = 100$ and $p = 80\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	❽	❸	⑥	-
Hier-C	⑦	❸	❽	5	❺	-
Hier-S	2	8	⑨	②	10	-
K-means-R	⑦	④	⑦	❸	⑥	-
K-means-A	⑦	❸	❽	❸	⑥	-
K-means-C	❻	❸	❽	④	⑥	-
K-means-S	⑦	5	⑨	②	7	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.17: A summary of the precision results for `Consensus` with $H = 500$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	1.5×10^6	-	2.4×10^6	2.0×10^6	5.9×10^7	-
Hier-C	1.5×10^6	-	2.3×10^6	1.6×10^6	5.9×10^7	-
Hier-S	1.6×10^6	-	1.5×10^6	1.6×10^6	5.8×10^7	-
K-means-R	1.5×10^6	-	3.4×10^6	2.7×10^6	4.7×10^7	-
K-means-A	1.8×10^6	-	3.4×10^6	2.0×10^6	7.9×10^7	-
K-means-C	1.0×10^6	-	2.4×10^6	2.0×10^6	7.5×10^7	-
K-means-S	1.3×10^6	-	2.5×10^6	2.8×10^6	4.8×10^7	-

Table 7.18: A summary of the timing results for `Consensus` with $H = 500$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. For the Leukemia dataset, the timing experiments are not reported because incomparable with those of the remaining datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	❽	❸	❺	-
Hier-C	⑦	❸	❽	❸	⑥	-
Hier-S	2	8	⑨	②	10	-
K-means-R	⑦	④	❽	6	⑥	-
K-means-A	⑦	❸	❽	5	⑥	-
K-means-C	❻	❸	⑨	5	⑥	-
K-means-S	⑦	8	10	②	⑥	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.19: A summary of the precision results for `Consensus` with $H = 250$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	3.8×10^5	-	8.0×10^5	1.1×10^6	3.7×10^7	-
Hier-C	3.8×10^5	-	8.2×10^5	1.1×10^6	3.7×10^7	-
Hier-S	7.9×10^5	-	8.2×10^5	7.0×10^5	3.7×10^7	-
K-means-R	5.2×10^5	-	1.4×10^6	1.4×10^6	3.1×10^7	-
K-means-A	5.7×10^5	-	1.2×10^6	1.2×10^6	4.8×10^7	-
K-means-C	5.3×10^5	-	1.2×10^6	1.3×10^6	4.4×10^7	-
K-means-S	6.2×10^5	-	1.2×10^6	1.1×10^6	5.1×10^7	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.20: A summary of the timing results for `Consensus` with $H = 250$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. For the Leukemia dataset, the timing experiments are not reported because incomparable with those of the remaining datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	8	❸	❽	❸	❺	-
Hier-C	⑦	❸	❽	④	⑥	-
Hier-S	2	8	⑨	9	2	-
K-means-R	⑦	④	❽	6	⑥	-
K-means-A	⑦	❸	❽	5	⑥	-
K-means-C	⑦	❸	❽	5	⑥	-
K-means-S	⑦	8	10	②	⑥	-

Table 7.21: A summary of the precision results for `Consensus` with $H = 100$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	5.9×10^4	-	9.6×10^4	4.3×10^4	1.0×10^6	-
Hier-C	4.2×10^4	-	4.9×10^4	4.9×10^5	1.0×10^6	-
Hier-S	4.8×10^4	-	5.1×10^4	5.3×10^5	1.6×10^6	-
K-means-R	5.2×10^5	-	1.0×10^6	9.6×10^5	1.2×10^7	-
K-means-A	5.6×10^5	-	8.7×10^5	6.8×10^5	2.5×10^6	-
K-means-C	5.5×10^5	-	5.0×10^5	7.8×10^5	1.0×10^7	-
K-means-S	3.5×10^5	-	5.6×10^5	8.0×10^5	1.5×10^7	-

Table 7.22: A summary of the timing results for `Consensus` with $H = 100$ and $p = 66\%$, on all algorithms, except NMF, and for the `Benchmark 1` datasets. For the Leukemia dataset, the timing experiments are not reported because incomparable with those of the remaining datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
Hier-A	0.190350	0.919174	0.498265	0.430841	0.523873
Hier-C	0.176446	0.676293	0.414214	0.483664	0.492815
Hier-S	0.000134	0.507680	0.161798	-0.01777	0.002036

Table 7.23: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by `Consensus` in Table 7.11 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
Hier-A	0.190350	0.919174	0.498265	0.430841	0.522578
Hier-C	0.237957	0.676293	0.414214	0.483664	0.555979
Hier-S	0.000134	-0.040230	0.161798	-0.009141	0.002036

Table 7.24: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by `Consensus` in Table 7.13 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

Index (defined in Section 2.1.1). In view of the results reported in the previous section, only the cases $H = 500, 250$ and $p = 80\%$ are discussed here. The corresponding results are reported in Tables 7.23 and 7.24, respectively. The relevant values of k^* are taken from Tables 7.11 and 7.13, respectively. The interested reader will find, at the following supplementary material web site [4], all the complete tables, in the Tables section, as Tables TS1-TS6. The second experiment follows the same lines as the first, but the clustering algorithm uses a Euclidean distance matrix. The results are reported in Table 7.25. In this case, the relevant values of k^* are taken from Table 7.11.

Tables 7.23 and 7.24 confirm the indication about the proper `Consensus` parameter setting identified in the previous section. Moreover, it is worth pointing out that there is no substantial difference between the results reported in Tables 7.23 and 7.25. Combining those results with the analogous ones obtained by Monti et al., one has an indication that the consensus matrix is at least as good as an Euclidean distance matrix, when used as input to hierarchical clustering algorithms.

	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
Hier-A	0.190350	0.910081	0.498265	0.430841	0.558884
Hier-C	0.135778	0.676293	0.414214	0.483664	0.413154
Hier-S	0.000134	0.507680	0.161798	-0.01777	0.002036

Table 7.25: For each dataset and each hierarchical algorithm considered here, the Euclidean distance matrix and number of clusters k^* predicted by `Consensus` in Table 7.11 is taken. That matrix is used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

7.5 An Approximation of Consensus: FC

In this section an approximation of `Consensus` is provided. This speedup is referred to as `FC` (Fast Consensus). Intuitively, a large number of clustering solutions, each obtained via a sample of the original dataset, seem to be required in order to identify the correct number of clusters. However, there is no theoretic reason indicating that those clustering solutions must each be generated from a *different* sample of the input dataset, as `Consensus` does. Based on this observation, this thesis proposes to perform, first, a sampling step to generate a data matrix D_1, which is then used to generate all clustering solutions for k in the range $[2, k_{max}]$. In terms of code, that implies a simple switch of the two iteration cycles of the `Consensus` procedure (see Chapter 3). Indeed, with reference to the stability measures discussed in Chapter 3, it is also worth noticing that each of the $k_{max} \times H$ clustering solutions needed is computed from a distinct dataset. As discussed here, this leads to inefficiencies, in particular in regard to agglomerative clustering algorithms, such as the hierarchical ones. Indeed, their ability to quickly compute a clustering solution with k clusters from one with $k+1$, typical of these methods, cannot be used within `Consensus` because, for each k, the dataset changes. The same holds true for divisive methods. In turn, that switch allows to obtain a speedup since costly computational duplications are avoided when the clustering algorithm C_1 is hierarchical. Indeed, once the switch is done, it becomes possible to interleave the computation of the measure with the level bottom-up construction of the hierarchical tree underlying the clustering algorithms. Specifically, only one dendogram construction is required rather than the repeated and partial construction of dendograms as in the `Consensus` procedure. Therefore, one uses, in full, the main characteristic of agglomerative algorithms briefly discussed in the section regarding `Consensus`. `FC` is formalized by the procedure given in Fig. 7.7. It is also worth pointing out that this switch is possible for several of the stability based measures detailed in Chapter 3. This general approximation paradigm is formalized by the procedure given in Fig. 7.8, where the macro operations and inputs are the same used for the STABILITY_MEASURE procedure detailed in Chapter 3. The "rule of thumb" that one uses to predict k^*, via `FC`, is the same as for `Consensus`. An example is reported in Fig. 7.4(b). It is worth pointing out that both the CDFs and Δ curve shapes for `FC` closely track those of the respective curves for `Consensus` Fig. 7.4(a).

7.5.1 FC and Its Parameters

In this section, the results of the experiments obtained with `FC` are reported and discussed. In analogy with `Consensus`, its precision and time performances depend on H and p. In order to compare the two measures along the parameters of interest, one uses, for `FC`, the same experimental setup detailed in the previous section for `Consensus`. Moreover, based on the results of the previous section, the discussion here is based only on the experiments for `FC` with $H = 250$ and $p = 80\%$.

It is worthy to anticipate that the results in this section will show that `FC` is a very good approximation of `Consensus` both as an internal validation measure and as a preprocessor for clustering algorithms. Remarkably, it is at least one order of magnitude faster in time when used in conjunction with hierarchical clustering algorithms or with partitional algorithms with a hierarchical initialization.

As pointed out in Chapter 5 in order to perform a better comparison between `Consensus` and its approximation, both `Benchmark 1` and `Benchmark 2` datasets are taken in account (see Section 5.1 for details) in this section.

7.5.1.1 FC as an Internal Validation Measure

Tables 7.26 and 7.27 report the results regarding `FC` as an internal validation measure for the `Benchmark 1` datasets. For this discussion, they are compared with the `Consensus` results reported in Tables 7.13 and 7.14. The interested reader will find, at the following supplementary material web site [4], all the complete tables as Tables TI7-TI12 for `FC` and the corresponding figures as Figs. S65-S139 in the Tables and Figures section, respectively. The time for the PBM dataset with $p = 66\%$ in the corresponding table is not reported, since it does not provide any relevant information.

Note that, in terms of precision, `FC` and `Consensus` provide nearly identical predictions on the CNS Rat and Yeast datasets, while their predictions are quite close on the Leukemia dataset. Moreover, in terms of time, note that `FC` is faster then `Consensus` by at least one order of magnitude on all hierarchical algorithms and K-means-A, K-means-C and K-means-S. In particular, `FC` is able to complete execution on the PBM dataset, as opposed to `Consensus`, with all of the mentioned algorithms. It is also worthy of notice that K-means-C also provides, for that dataset, a reasonable estimate of the number of clusters present in it. Another point of interest is the performance of `FC` with K-means-R since the algorithm engineering used in its implementation grants good results on the largest datasets used with that clustering algorithm.

It is somewhat unfortunate, however, that those quite substantial speedups have only minor effects when one uses `NMF` as a clustering algorithm, which is a clear indication that the time taken by `NMF` to converge to a clustering solution accounts for most of the time performance of `FC` in that setting, in analogy with `Consensus`.

As for `Benchmark 2` datasets, both `Consensus` and `FC` are computed for a number of cluster values in the range $[2, 30]$. The prediction value, k^*, is based on the plot of the $\Delta(k)$ curve (defined in Chapter 3) as indicated in [62]. The corresponding plots are available at the following supplementary material web site [4], in the Figures section, as Figs. M1-M12 and M13-M24 for `Consensus` and `FC`, respectively. The corresponding results are reported in Table 7.30 and Table 7.33 for the simulated datasets, while the corresponding results for the microarray datasets are in Tables 7.28-7.29 and Tables 7.31-7.32 for `Consensus` and `FC`, respectively.

By comparing the results in the mentioned tables, it is of great interest to notice that, on the datasets in `Benchmark 2`, there is no difference whatsoever in the predictions between `Consensus` and `FC`. Even more remarkably, by analyzing the Δ curves from which the predictions are made (see Methods section), one discovers that the ones produced by `Consensus` and `FC` are nearly identical (see again Figs. M1-M24 at the following supplementary material web site [4]). However, on the microarray datasets on `Benchmark 2`, `FC` is at least one order of magnitude faster than `Consensus`, with exactly the same algorithms indicated for the `Benchmark 1` datasets. `NMF` results to be problematic also on the datasets on `Benchmark 2`.

It is of some interest to point out that, as detailed in the previous section, `FC` builds the same number of connectivity matrices as `Consensus`. However, it uses only H "new" matrices, each sampled from the input dataset, rather than $H \times k$ "new" matrices as `Consensus` does. Adding this observation to the ones of the preceding subsection, one understands that the number of connectivity matrices computed by `FC` is key to its precision performance, again in analogy with `Consensus`. The novelty, by far non-obvious, is that those matrices can be computed by taking a relatively small number of samples from the input matrix. Moreover, Figs. 7.5 and 7.6 provides the Δ curve both for `Consensus` and `FC` for $p = 80\%$ and different values of H, in order to show how the behavior of the two curves is practically identically for a $H > 100$. From these figures it is possible to see how `FC` preserves the same

	Precision					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	⑦	❸	❽	❸	❺	2
Hier-C	❻	④	❽	5	⑥	14 - ⑰
Hier-S	2	8	❽	②	10	2
K-means-R	❻	④	⑦	④	⑥	16
K-means-A	⑦	❸	❽	❸	⑥	12
K-means-C	❻	④	❽	④	⑥	12
K-means-S	❻	7	⑨	②	⑥	2
NMF-R	❻	④	⑦	④	-	-
NMF-A	⑦	❸	⑦	❸	-	-
NMF-C	❻	❸	❽	④	-	-
NMF-S	2	8	⑨	②	-	-
Gold solution	**6**	**3**	**8**	**3**	**5**	**18**

Table 7.26: A summary of the precision results for FC with $H = 250$ and $p = 80\%$, on all algorithms, and for the Benchmark 1 datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

outstanding properties of Consensus and that for a reasonable value of H the precision of the measures is the same (see Fig. 7.6 for $H = 250$).

7.5.1.2 FC and Similarity Matrices

The same experiments described for the evaluation of Consensus for the computation of a similarity matrix have been performed here for FC. For the presentation of the results, the same organization of Section 7.4.2 is followed here, i.e., the results for the Benchmark 1 datasets are presented and discussed first. Indeed, the experiments reported in Table 7.35 are the same as the ones reported in Table 7.24 for Consensus. Again, there is no difference between the two tables. Therefore, also in this case, FC is a good approximation of Consensus. For completeness, we report that, for FC on the Benchmark 1 datasets, the interested reader will find, at the following supplementary material web site [4], all the complete tables, in the Tables section, as Tables TS7-TS12, for each experimental setup.

As for Benchmark 2 datasets, the results of the experiments, for the microarrays datasets, are reported in Tables 7.34 and 7.36 for Consensus and FC, respectively. Tables 7.38 and 7.39 report the results for the simulated datasets for Consensus and FC, respectively. Also for the Benchmark 2 datasets, there is no difference between the two methods.

In analogy with Consensus and the Benchmark 1 datasets, the clustering results obtained with the use of the similarity matrices computed by Consensus and FC are compared, for the Benchmark 2 datasets, against the clustering results obtained with the use of Euclidean distance. The relevant values of k^* are taken from Table 7.28 for Consensus and Table 7.31 for FC. The results are reported in Tables 7.36-7.37 and Tables 7.38-7.40. They confirm that the consensus matrix is at least as good as an Euclidean distance matrix, when used as input to hierarchical clustering algorithms, even when computed by FC.

	Timing					
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	5.9×10^4	2.7×10^4	7.0×10^4	6.8×10^4	1.5×10^6	4.2×10^7
Hier-C	5.9×10^4	2.7×10^4	6.5×10^4	6.7×10^4	1.4×10^6	3.4×10^7
Hier-S	8.1×10^4	2.7×10^4	5.8×10^4	6.2×10^4	2.2×10^6	4.4×10^7
K-means-R	3.7×10^5	3.7×10^5	1.2×10^6	1.1×10^6	1.6×10^7	1.6×10^8
K-means-A	3.1×10^5	2.0×10^5	9.3×10^5	9.0×10^5	1.8×10^7	2.1×10^8
K-means-C	2.5×10^5	6.0×10^5	6.5×10^5	9.4×10^5	1.4×10^7	2.0×10^8
K-means-S	3.7×10^5	5.8×10^5	6.9×10^5	9.4×10^5	1.9×10^7	2.4×10^8
NMF-R	1.1×10^8	1.3×10^7	6.3×10^7	7.5×10^7	-	-
NMF-A	3.0×10^7	4.0×10^6	1.2×10^7	1.6×10^7	-	-
NMF-C	2.9×10^7	4.0×10^6	1.2×10^7	1.6×10^7	-	-
NMF-S	3.5×10^7	4.0×10^6	1.2×10^7	1.5×10^7	-	-

Table 7.27: A summary of the timing results for `FC` with $H = 250$ and $p = 80\%$, on all algorithms, and for the `Benchmark 1` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision		
	Novartis	St.Jude	Normal
Hier-A	⑤ - 6	❻	10
Hier-C	❹ - ⑤	⑤ - ❻	10
Hier-S	⑤	2	10
K-means-R	⑤	❻	10
K-means-A	⑤ - 6	❻	8
K-means-C	❹ - ⑤	⑤ - ❻	10
K-means-S	⑤	❻	10
NMF-R	-	-	-
NMF-A	-	-	-
NMF-C	-	-	-
NMF-S	-	-	-
Gold solution	**4**	**6**	**13**

Table 7.28: A summary of the precision results for `Consensus` with $H = 250$ and $p = 80\%$, on all algorithms and for the `Benchmark 2` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing		
	Novartis	St.Jude	Normal
Hier-A	1.0×10^7	3.7×10^7	9.5×10^6
Hier-C	1.0×10^7	3.7×10^7	9.2×10^6
Hier-S	9.8×10^6	3.7×10^7	9.4×10^6
K-means-R	1.8×10^7	1.5×10^7	6.3×10^6
K-means-A	1.4×10^7	6.8×10^7	1.1×10^7
K-means-C	1.5×10^7	6.8×10^7	1.0×10^7
K-means-S	1.6×10^7	6.8×10^7	1.1×10^7
NMF-R	-	-	-
NMF-A	-	-	-
NMF-C	-	-	-
NMF-S	-	-	-

Table 7.29: A summary of the timing results for `Consensus` with $H = 250$ and $p = 80\%$, on all algorithms and for the `Benchmark 2` datasets. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision		
	Gaussian3	Gaussian5	Simulated6
Hier-A	❸	❺	⑤
Hier-C	❸	❺	⑤
Hier-S	②	2	⑦
K-means-R	❸	❺	❻
K-means-A	❸	❺	⑤
K-means-C	❸	❺	⑤
K-means-S	②	❺	❻
Gold solution	**3**	**5**	**6**

Table 7.30: A summary of the precision results for `Consensus` with $H = 250$ and $p = 80\%$, on all algorithms, except NMF, and for the simulated datasets in `Benchmark 2`.

	Precision		
	Novartis	St.Jude	Normal
Hier-A	⑤ - 6	❻	10
Hier-C	❹ - ⑤	⑤ - ❻	10
Hier-S	⑤	2	10
K-means-R	⑤	❻	10
K-means-A	⑤ - 6	❻	8
K-means-C	❹ - ⑤	⑤ - ❻	10
K-means-S	⑤	❻	10
NMF-R	-	-	-
NMF-A	-	-	-
NMF-C	-	-	-
NMF-S	-	-	-
Gold solution	**4**	**6**	**13**

Table 7.31: A summary of the precision results for `FC` with $H = 250$ and $p = 80\%$, on all algorithms and for the microarray datasets in `Benchmark 2`. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Timing		
	Novartis	St.Jude	Normal
Hier-A	4.0×10^5	1.6×10^6	3.4×10^5
Hier-C	3.9×10^5	1.4×10^6	3.3×10^5
Hier-S	4.4×10^5	1.5×10^6	3.4×10^5
K-means-R	1.4×10^7	5.9×10^6	2.0×10^6
K-means-A	5.5×10^6	3.2×10^7	5.4×10^6
K-means-C	6.5×10^6	3.2×10^7	2.1×10^6
K-means-S	7.8×10^6	4.9×10^7	2.1×10^6
NMF-R	-	-	-
NMF-A	-	-	-
NMF-C	-	-	-
NMF-S	-	-	-

Table 7.32: A summary of the precision results for FC with $H = 250$ and $p = 80\%$, on all algorithms and for the microarray datasets in Benchmark 2. Cells with a dash indicate that the experiments were terminated due to their high computational demand.

	Precision		
	Gaussian3	Gaussian5	Simulated6
Hier-A	❸	❺	⑤
Hier-C	❸	❺	⑤
Hier-S	②	2	⑦
K-means-R	❸	❺	❻
K-means-A	❸	❺	⑤
K-means-C	❸	❺	⑤
K-means-S	②	❺	❻
Gold solution	**3**	**5**	**6**

Table 7.33: A summary of the precision results for FC with $H = 250$ and $p = 80\%$, on all algorithms, except NMF, and for the simulated datasets in Benchmark 2.

	Novartis	St.Jude	Normal
Hier-A	0.641611	0.173717	0.572747
Hier-C	0.515570	0.438039	0.521355
Hier-S	0.320264	$-7.88788e^{-4}$	0.502043

Table 7.34: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by Consensus in Table 7.28 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the Adjusted Rand Index.

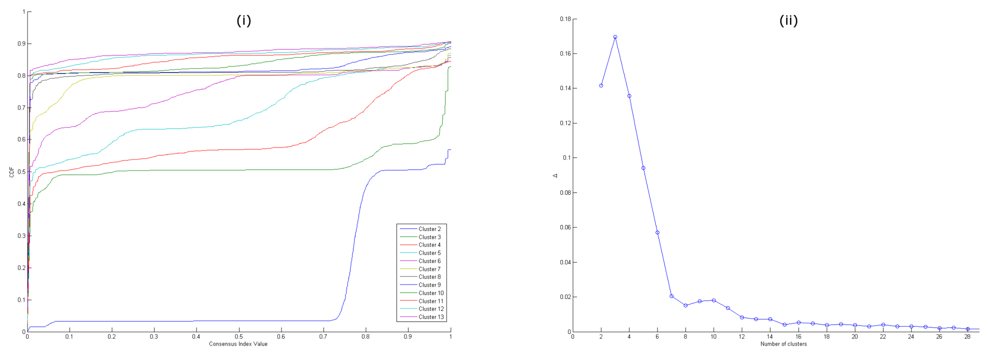

Figure 1(a) - An example of number of cluster prediction with the use of Consensus with H=250 and p=80%. The experiment is derived from the NCI60 dataset, with use of the Hier-A clustering algorithm. The plots of the CDF curves is shown in (i), yielding a monotonically increasing value as a function of k. The plot of the Δ curve is shown in (ii), where the flattening effect corresponding to k* is evident for k ≥ k* = 8.

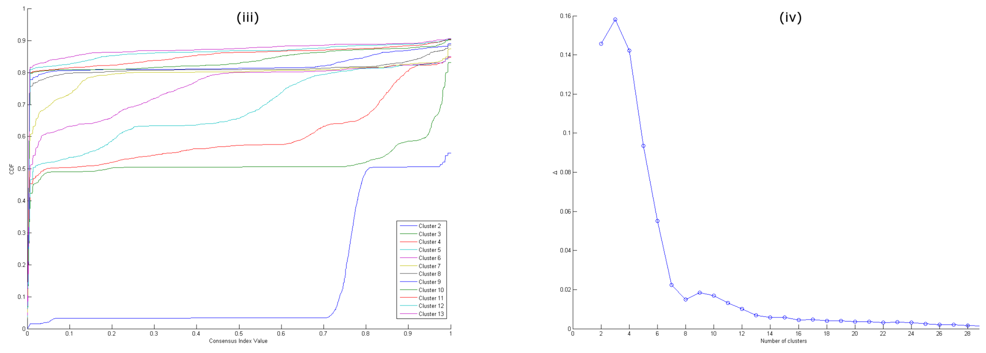

Figure 1(b) - An example of number of cluster prediction with the use of FC with H=250 and p=80%. The experiment is derived from the NCI60 dataset, with use of the Hier-A clustering algorithm. The plots of the CDF curves is shown in (iii), yielding a monotonically increasing value of A as a function of k. The plot of the Δ curve is shown in (iv), where the flattening effect corresponding to k* is evident for k ≥ k* = 8.

Figure 7.4: The experiment is derived from the NCI60 dataset, with the use of the Hier-A clustering algorithm. (a) Figure for `Consensus` with $H = 250$ and $p = 80\%$: the plot of the CDF curves is shown (i), yielding a monotonically increasing value of A as a function of k. The plot of the Δ curve is shown in (ii), where the flattening effect corresponding to k^* is evident for $k \geq k^* = 8$. (b) Figure for `FC` with $H = 250$ and $p = 80\%$: the plots of the CDF curves is shown in (iii), yielding a monotonically increasing value of A as a function of k. The plot of the Δ curve is shown in (iv), where the flattening effect corresponding to k^* is evident for $k \geq k^* = 8$.

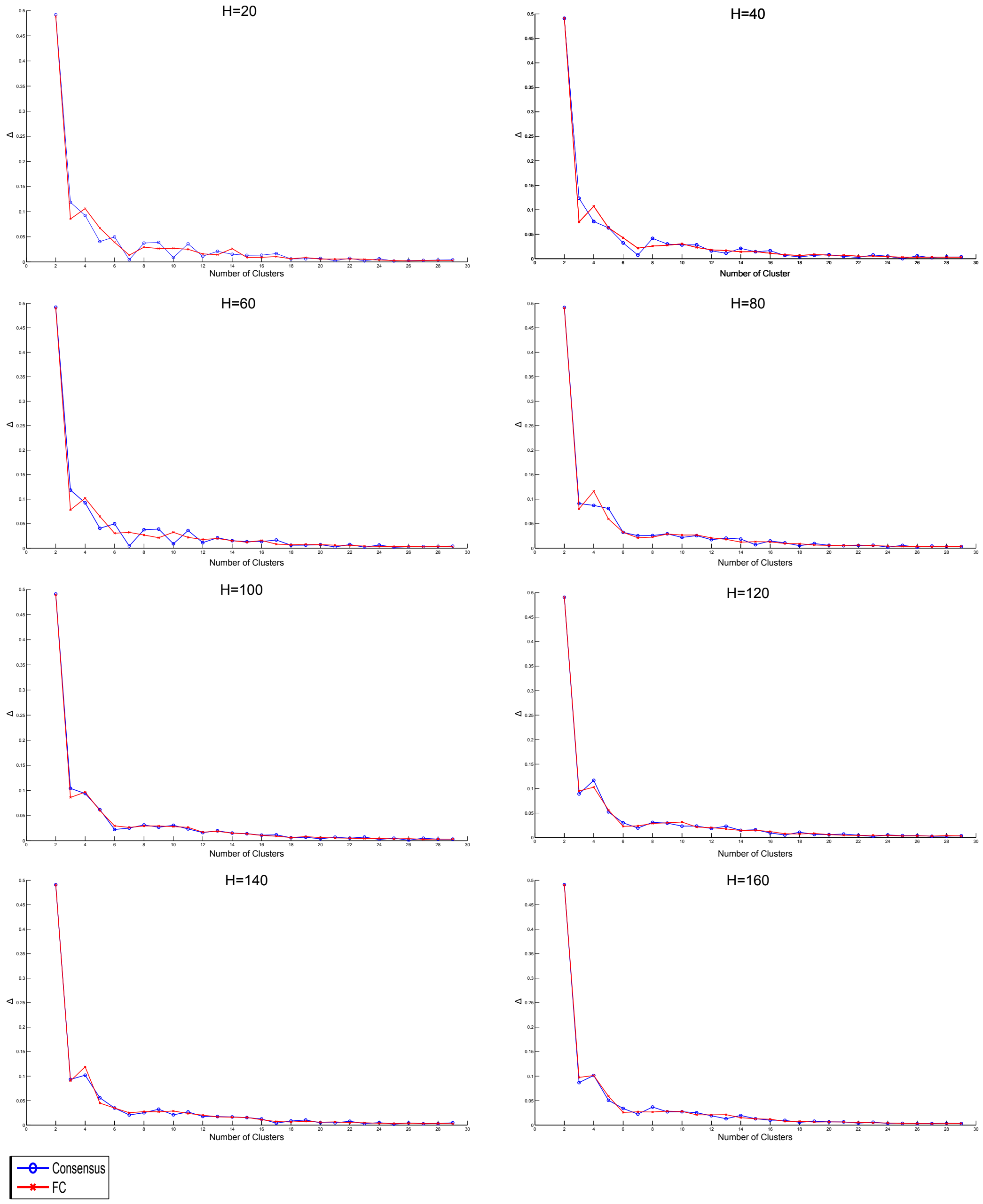

Figure 7.5: Plot of the Δ curves for `Consensus` and `FC` with p=80% and H=20,40,60,80,100,120,140,160. The experiment is derived from the Lymphoma dataset, with the use of the Hier-A clustering algorithm.

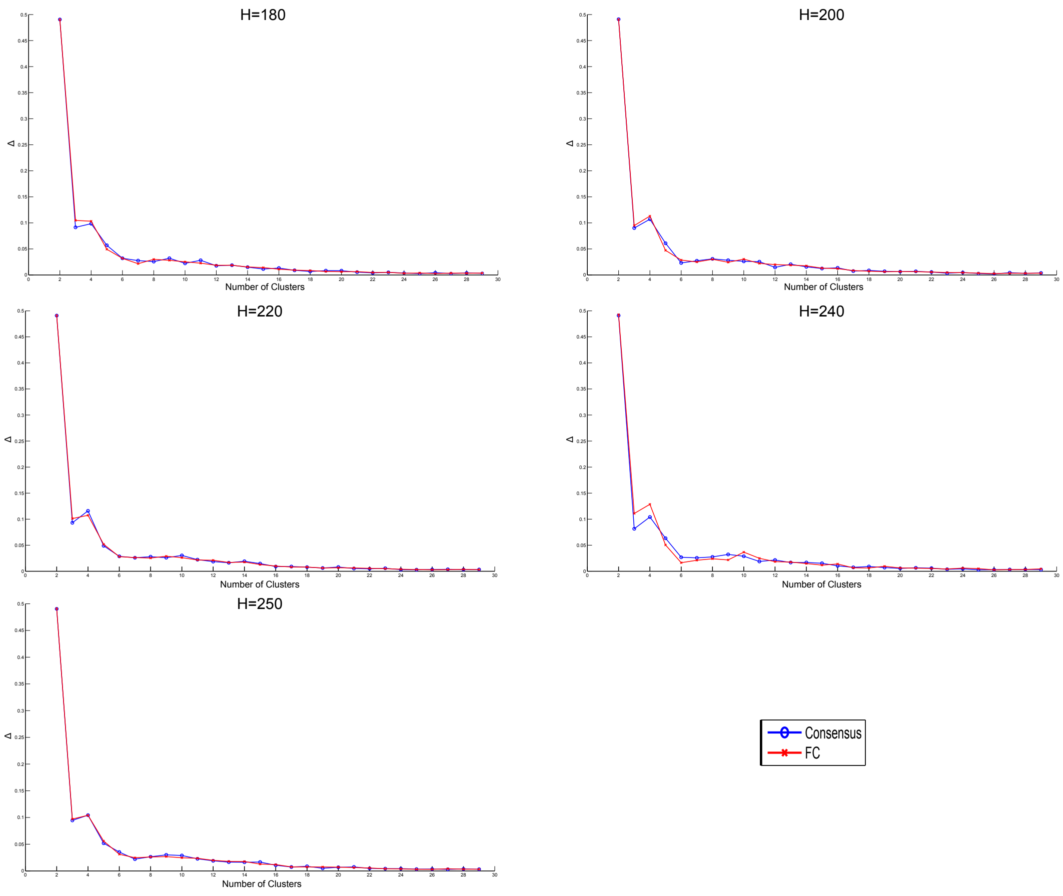

Figure 7.6: Plot of the Δ curves for `Consensus` and `FC` with p=80% and H=180,200,220,250. The experiment is derived from the Lymphoma dataset, with the use of the Hier-A clustering algorithm.

```
FC($H_c$, $< C_1 >$, $D$, $k_{max}$)

for $i \leftarrow 1$ to $H_c$ do
  begin
1.    Generate (via a subsampling) a data matrix $D_i$
    for $k \leftarrow 2$ to $k_{max}$ do
      begin
2.      Let $P_1$ be the partition of $D_i$ into $k$ clusters with the use of $C_1$
3.      Based on $P_1$ compute the connectivity matrix $M_i^k$
      end
  end
for $k \leftarrow 2$ to $k_{max}$ do
  begin
4.  Compute the consensus matrix $\mathcal{M}^k$
  end
5. Based on the $k_{max} - 1$ consensus matrices, return a prediction for $k^*$
```

Figure 7.7: The FC procedure

```
FAST_STABILITY_MEASURE($k_{min}, k_{max}, D, H, \alpha, \beta, < C_1, C_2, \ldots, C_t >$)

while $H$ do
  begin
1.  $< D_1, D_2, \ldots, D_l > \leftarrow < \mathrm{DGP}(D_0, \beta), \mathrm{DGP}(D_0, \beta), \ldots, \mathrm{DGP}(D_0, \beta) >$
   for $k \leftarrow k_{min}$ to $k_{max}$ do
     begin
2.     $< D_{T,0}, D_{T,1}, \ldots, D_{T,l}, D_{L,0}, D_{L,1}, \ldots, D_{L,l} > \leftarrow$ SPLIT($< D_0, D_1, \ldots, D_l >, \alpha$)
3.     $< G > \leftarrow$ ASSIGN($< D_{T,0}, D_{T,1}, \ldots, D_{T,l} >, < C_1, C_2, \ldots, C_t >$)
4.     $< C_{i_1}, C_{i_2}, \ldots, C_{i_q} > \leftarrow$ TRAIN($< G >$)
5.     $< \hat{G} > \leftarrow$ ASSIGN($< D_{L,0}, D_{L,1}, \ldots, D_{L,l} >, < C_1, C_2, \ldots, C_t >$)
6.     $< P_1, P_2, \ldots, P_z > \leftarrow$ CLUSTER($\hat{G}, k$)
7.     $u \leftarrow$ COLLECT_STATISTIC($< P_1, P_2, \ldots, P_z >$)
8.     $S^k \leftarrow S^k \bigcup \{u\}$
     end
  end
 for $k \leftarrow k_{min}$ to $k_{max}$ do
   begin
9.    $R^k \leftarrow$ SYNOPSIS($S^k$)
   end
10. $k^* \leftarrow$ SIGNIFICANCE_ANALYSIS($R^{k_{min}}, \ldots, R^{k_{max}}$)
return $(k^*)$
```

Figure 7.8: The FAST_STABILITY_MEASURE procedure.

	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast	PBM
Hier-A	0.190350	0.919174	0.498265	0.430841	0.528360	0.000261
Hier-C	0.176446	0.676293	0.414214	0.483664	0.589179	0.672256
Hier-S	0.000134	0.507680	0.171124	-0.009141	0.00203	0.000261

Table 7.35: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by FC in Table 7.26 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the Adjusted Rand Index.

	Novartis	St.Jude	Normal
Hier-A	0.641611	0.173717	0.572747
Hier-C	0.515570	0.435431	0.537849
Hier-S	0.320264	-7.88788e^{-4}	0.502043

Table 7.36: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by FC in Table 7.31 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the Adjusted Rand Index.

7.5.2 Comparison of FC with other Internal Validation Measures

It is also of interest to compare FC with other validation measures that are available in the Literature. One takes, as reference, the benchmarking results reported in Chapter 6, since both the datasets and the experimental setup are identical to the ones used here. It is worth pointing out that this benchmark show that there is a natural hierarchy, in terms of time, for the measures taken in account. Moreover, the faster the measure, the less accurate it is. From that study and for completeness, taking in account Tables 6.15 and 6.16 of Section 6.3 one reports in Tables 7.41 and 7.42 the best performing measures, with the addition of FC and the other "best" approximations proposed in this chapter. From that table, one extract and report, in Tables 7.43 and 7.44, the fastest and best performing measures - again, with the addition of FC. As is self-evident from that latter table, FC with Hier-A is within a one order of magnitude difference in speed with respect to the fastest measures, i.e., WCSS and G-Gap. Quite remarkably, it grants a better precision in terms of its ability to identify the

	Novartis	St.Jude	Normal
Hier-A	0.544647	0.16992	0.57274
Hier-C	0.51557	0.42637	0.52135
Hier-S	0.32026	$-7.88786e^{-4}$	0.50204

Table 7.37: For each dataset and each hierarchical algorithm considered here, the Euclidean distance matrix and number of clusters k^* predicted by Consensus in Table 7.28 is taken. That matrix is used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the Adjusted Rand Index.

	Gaussian3	Gaussian5	Simulated6
Hier-A	1.0	0.85942	−0.59990
Hier-C	1.0	0.82638	−0.59990
Hier-S	0.0	0.0	−0.19586

Table 7.38: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by `Consensus` in Table 7.30 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

	Gaussian3	Gaussian5	Simulated6
Hier-A	1.0	0.859429	−0.59990
Hier-C	1.0	0.830103	−0.59990
Hier-S	0.0	0.0	−0.19586

Table 7.39: For each dataset and each hierarchical algorithm considered here, the consensus matrix corresponding to the number of clusters k^* predicted by `FC` in Table 7.33 is taken. That matrix is transformed into a distance matrix, which is then used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

	Gaussian3	Gaussian5	Simulated6
Hier-A	1.0	0.82729	-0.59990
Hier-C	1.0	0.65218	-0.59990
Hier-S	0.0	0.0	-0.195867

Table 7.40: For each dataset and each hierarchical algorithm considered here, the Euclidean distance matrix and number of clusters k^* predicted by `Consensus` in Table 7.30 is taken. That matrix is used by the clustering algorithm to produce k^* clusters. The agreement of that clustering solution with the gold solution of the given dataset is measured via the `Adjusted Rand` Index.

	Precision				
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
`WCSS`-K-means-C	⑤	❸	❽	8	④
`WCSS`-R-R0	⑤	④	❽	❸	④
`G-Gap`-K-means-R	⑦	❸	4	④	⑥
`G-Gap`-R-R5	⑤	④	2	②	④
`FOM`-K-means-C	⑦	8	❽	④	④
`FOM`-K-means-S	❻	❸	❽	8	④
`FOM`-R-R5	❻	❸	⑦	5	❺
`FOM`-Hier-A	⑦	❸	⑦	6	⑥
`DIFF-FOM`-K-means-C	⑦	❸	⑦	④	3
`FC`-Hier-A	⑦	❸	❽	❸	❺
`FC`-Hier-C	❻	④	❽	5	⑥
`FC`-K-means-R	❻	④	⑦	④	⑥
`FC`-K-means-A	⑦	❸	❽	❸	⑥
`FC`-K-means-C	❻	❸	❽	④	⑥
`FC`-K-means-S	⑦	④	10	②	⑥
`Clest-F`-K-means-R	❻	❸	15	②	④
`Clest-FM`-K-means-R	8	④	❽	②	④
`Consensus`-Hier-A	⑦	❸	❽	❸	❺
`Consensus`-Hier-C	❻	④	❽	5	⑥
`Consensus`-K-means-R	❻	④	⑦	❸	⑥
`Consensus`-K-means-A	⑦	❸	❽	❸	⑥
`Consensus`-K-means-C	❻	❸	❽	④	⑥
`Consensus`-K-means-S	⑦	④	10	②	⑥
Gold solution	**6**	**3**	**8**	**3**	**5**

Table 7.41: A summary of precision results of the best performing measures taken into account in Chapter 6, with the addition of `WCSS`-R, `G-Gap`, `FOM`-R, `DIFF-FOM` and `FC` with $H = 500$ and $p = 80\%$.

underlying structure in each of the benchmark datasets. It is also of relevance to point out that `FC` with Hier-A has a time performance comparable to that of `FOM`, but again it has a better precision performance. Notice that, none of the three just-mentioned measures depends on any parameter setting, implying that no speedup will result from a tuning of the algorithms.

The results outlined above are particularly significant since (i) `FOM` is one of the most established and highly-referenced measures specifically designed for microarray data; (ii) in purely algorithmic terms, `WCSS` and `G-Gap`, are so simple as to represent a “lower bound” in terms of the time performance that is achievable by any data-driven internal validation measure. In conclusion, the experiments reported here show that `FC` is quite close in time performance to three of the fastest data-driven validation measures available in the Literature, while also granting better precision results. In view of the fact that the former measures are considered reference points in this area, the speedup of `Consensus` proposed here seems to be a non-trivial step forward in the area of data-driven internal validation measures.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
WCSS-K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
WCSS-R-R0	1.2×10^3	8.0×10^2	4.1×10^3	3.0×10^3
G-Gap-K-means-R	2.4×10^3	2.0×10^3	8.3×10^4	8.4×10^3
G-Gap-R-R5	1.2×10^3	8.0×10^2	4.5×10^4	3.2×10^3
FOM-K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
FOM-K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5
FOM-R-R5	3.9×10^3	3.7×10^4	2.1×10^5	7.6×10^4
FOM-Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
DIFF-FOM-K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
FC-Hier-A	4.7×10^4	3.5×10^4	5.2×10^4	1.3×10^5
FC-Hier-C	4.4×10^4	2.7×10^4	1.3×10^5	1.3×10^5
FC-K-means-R	7.2×10^5	7.7×10^5	2.5×10^6	2.3×10^6
FC-K-means-A	5.6×10^5	4.2×10^5	1.1×10^6	1.5×10^6
FC-K-means-C	5.4×10^5	4.8×10^5	1.1×10^6	1.0×10^6
FC-K-means-S	7.7×10^5	4.7×10^5	1.3×10^6	1.1×10^6
Clest-F-K-means-R	1.2×10^6	-	-	-
Clest-FM-K-means-R	1.2×10^6	-	-	-
Consensus-Hier-A	9.2×10^5	7.9×10^5	2.0×10^6	1.9×10^6
Consensus-Hier-C	8.7×10^5	6.9×10^5	2.0×10^6	2.0×10^6
Consensus-K-means-R	1.0×10^6	1.3×10^6	3.4×10^6	3.0×10^6
Consensus-K-means-A	1.3×10^6	1.6×10^6	3.0×10^6	2.6×10^6
Consensus-K-means-C	1.3×10^6	1.8×10^6	2.9×10^6	2.6×10^6
Consensus-K-means-S	1.5×10^6	1.8×10^6	3.2×10^6	2.8×10^6

Table 7.42: A summary of the timing results best performing measures taken into account in Chapter 6, with the addition of WCSS-R, G-Gap, FOM-R, DIFF-FOM and FC with $H = 500$ and $p = 80\%$. Cell with a dash indicates that the experiment was performed on a smaller interval of cluster values with respect to CNS Rat and so the time performance are not comparable.

	Precision				
	CNS Rat	Leukemia	NCI60	Lymphoma	Yeast
WCSS-K-means-C	⑤	❸	❽	8	④
WCSS-R-R0	⑤	④	❽	❸	④
G-Gap-K-means-R	⑦	❸	4	④	⑥
G-Gap-R-R5	⑤	④	2	②	④
FOM-K-means-C	⑦	8	❽	④	④
FOM-K-means-S	❻	❸	❽	8	④
FOM-R-R5	❻	❸	⑦	5	❺
FOM-Hier-A	⑦	❸	⑦	6	⑥
DIFF-FOM-K-means-C	⑦	❸	⑦	④	3
FC-Hier-A	⑦	❸	❽	❸	❺
FC-Hier-C	❻	④	❽	5	⑥
Gold solution	**6**	**3**	**8**	**3**	**5**

Table 7.43: A summary of the precision results of best performing measures taken from the benchmarking of Chapter 6, with the addition of WCSS-R, G-Gap, FOM-R, DIFF-FOM and FC, with $H = 250$ and $p = 80\%$.

	Timing			
	CNS Rat	Leukemia	NCI60	Lymphoma
WCSS-K-means-C	1.7×10^3	1.3×10^3	5.0×10^3	4.0×10^3
WCSS-R-R0	1.2×10^3	8.0×10^2	4.1×10^3	3.0×10^3
G-Gap-K-means-R	2.4×10^3	2.0×10^3	8.3×10^4	8.4×10^3
G-Gap-R-R5	1.2×10^3	8.0×10^2	4.5×10^4	3.2×10^3
FOM-K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
FOM-K-means-S	2.9×10^4	1.0×10^5	7.1×10^5	3.6×10^5
FOM-R-R5	3.9×10^3	3.7×10^4	2.1×10^5	7.6×10^4
FOM-Hier-A	1.6×10^3	7.5×10^3	5.1×10^4	1.8×10^4
DIFF-FOM-K-means-C	1.9×10^4	9.4×10^4	5.5×10^5	2.6×10^5
FC-Hier-A	5.9×10^4	2.7×10^4	7.0×10^4	6.8×10^4
FC-Hier-C	5.9×10^4	2.7×10^4	6.5×10^4	6.7×10^4

Table 7.44: A summary of the time results of best performing measures taken from the benchmarking of Chapter 6, with the addition of WCSS-R, G-Gap, FOM-R, DIFF-FOM and FC, with $H = 250$ and $p = 80\%$.

Chapter 8

Conclusions and Future Directions

In this thesis, an extensive study of internal validation measures is proposed, with attention to the analysis of microarray data. In particular, this dissertation has contributed to the area as follows:

A Paradigm for Stability Measures. A new general paradigm of stability internal validation measures is proposed. It is also shown that each of the known stability based measures is an instance of such a novel paradigm. Surprisingly, also `Gap` falls within the new paradigm. Moreover, from this general algorithmic paradigm, it is simple to design new stability internal measure combining the building blocks of the detailed measures.

Benchmarking of Internal Validation Measures. A benchmarking of internal validation measures, taking into account both the precision and time, is proposed. This study provides further insights into the relative merits of each of the measures considered, from which more accurate and useful guidelines for their use can be inferred. In particular, when computer time is taken into account, there is a hierarchy of measures, with `WCSS` being the fastest and `Consensus` the slowest. Overall, `Consensus` results to be the method of choice. It is also to be stressed that no measure performed well on large datasets.

Fast Approximations. Based on the above benchmarking, the idea of extensions and approximations of internal validation measures has been systematically investigated. The resulting new measures turn out to be competitive, both in time and precision. In particular, `G-Gap` and `FC` an approximation of `Gap` and `Consensus`, respectively, are proposed. As it is evident from the results obtained, the overall performance of the approximations is clearly superior to the “original” measures. Moreover, depending on the dataset, they are at least one orders of magnitude faster. In terms of the existing Literature on data-driven internal validation measures, `FC` is only one order of magnitude away from the fastest measures, yet granting a superior performance in terms of precision. Although `FC` does not close the gap between the time performance of the fastest internal validation measures and the most precise, it is a substantial step forward towards that goal.

Benchmarking of NMF as a clustering algorithm. A benchmarking of NMF as a clustering algorithm on microarray data is proposed. Unfortunately, in view of the steep computational price one must pay, the use of NMF as a clustering algorithm does not seem to be justified. Indeed, NMF is at least two orders of magnitude slower than a classical clustering algorithm and with a worse precision.

Future Directions. This thesis suggests several interesting directions of investigation. Some of them are mentioned next:

- Techniques that would enhance the performance of NMF. In particular, a relevant issue is to compute a solution for k clusters starting from one with $k \pm 1$ clusters.

That is, an incremental/decremental version of NMF. Such a version could yield a substantial speedup when NMF is used as a clustering algorithm in conjunction with `Consensus` and `FC`.

- The intrinsic and relative study of stability validation measure generated from the stability paradigm, mixing the building blocks available today.
- The design of fast approximations of other stability internal validation measures.
- An internal validation measure that closes the gap between the time performance of the fastest internal validation measures and the most precise.
- A comparison among the best data driven validation measures discussed here and Bayesian method that solve the same problem, i.e., [141].

Bibliography

[1] Broad institute. http://www.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=89. 63

[2] NCI 60 Cancer Microarray Project. http://genome-www.stanford.edu/NCI60. 61

[3] Suppelementary material web site benchmarking. http://www.math.unipa.it/~raffaele/suppMaterial/benchmarking/benchmarking/Index.html. 74, 75, 77, 79, 82, 84, 93

[4] Supplementary material web site speedup. http://www.math.unipa.it/~utro/suppMaterial/speedUp/. 100, 106, 108, 109

[5] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66:671–687, 2003. 16

[6] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *STOC '06: Proceedings of the 38th annual ACM Symposium on Theory of computing*, pages 557–563. ACM, 2006. 16

[7] A.Langville, C. Meyer, and R. Albright. Initializations for the Nonnegative Matrix Factorization, 2006. 49

[8] A.A. Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C Boldrick, H. Sabet, T. Tran, X. Yu, J.I Powell, L. Yang, G.E. Marti, T. Moore, J. Jr Hudson, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C Byrd, D. Botstein, P.O. Brown, and L.M. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000. 1, 61

[9] N. Alon. Problems and results in extremal combinatorics - ii. *Discrete Mathematics*, 308:4460–4472, 2008. 16

[10] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96:6745–6750, 1999. 1

[11] R.B. Altman. Professor Donald Knuth on Bioinformatics. http://www-helix.stanford.edu/people/altman/bioinformatics.html. 1

[12] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51:117–122, 2008. 16

[13] L. Badea. Clustering and metaclustering with Nonnegative Matrix Decompositions. In *16th European Conference on Machine Learning*. Springer, 2005. 55

[14] A.J. Bell and T.J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37:3327–3338, 1997. 48

[15] S. Ben-David, U. von Luxburg, and D. Pál. A sober look at clustering stability. *Lecture Notes in Computer Science*, 4005:5, 2006. 28

[16] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustering data. In *Seventh Pacific Symposium on Biocomputing*, pages 6–17. ISCB, 2002. 15, 33, 99

[17] J. Benesty, D. Morgan, and M. Sondhi. A better understanding and an improved solution to the problems of stereophonic acoustic echo cancellation. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 1*, page 303. IEEE Computer Society, 1997. 15

[18] M.W. Berry and M. Browne. *Understanding search engines: mathematical modeling and text retrieval.* Society for Industrial and Applied Mathematics, 1999. 55

[19] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate Nonnegative Matrix Factorization. In *Computational Statistics and Data Analysis*, pages 155–173. Elsevier, 2006. 49, 52, 53

[20] A. Bertoni and G. Valentini. Randomized maps for assessing the reliability of patients clusters in DNA microarray data analyses. *Artificial Intelligence in Medicine*, 37:85–109, 2006. 15

[21] A. Bertoni and G. Valentini. Model order selection for bio-molecular data clustering. *BMC Bioinformatics*, 8, 2007. 35

[22] A. Bhattacharya, P. Kar, and M. Pal. On low distortion embeddings of statistical distance measures into low dimensional spaces. In *DEXA*, pages 164–172, 2009. 16

[23] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001. 16

[24] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola F, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, and V. Sondak. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406:536–540, 2000. 15

[25] H.H. Bock. On some significance tests in cluster analysis. *Journal of Classification*, 2:77–108, 1985. 11

[26] J.A. Bondy and U.S.R. Murty. *Graph Theory With Applications.* Elsevier Science Ltd, 1976. 6

[27] A. Borodin, R. Ostrovsky, and Y. Rabani. Subquadratic approximation algorithms for clustering problems in high dimensional space. *Machine Learning*, 56:153–167, 2004. 2

[28] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41:1350–1362, 2008. 49

[29] J.N. Breckenridge. Replicating cluster analysis: Method, consistency, and validity. *Multivariate Behavioral Research*, 24(2):147–161, 1989. 2, 28, 31

[30] J.-P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, 101:4164–4169, 2004. x, 39, 49, 56, 57, 60

[31] P. Carmona-Saez, R.D. Pascual-Marqui, F. Tirado, J.M. Carazo, and A. Pascual-Montano. Biclustering of gene expression data by non-smooth Non-negative Matrix Factorization. *BMC Bioinformatics*, 7:78, 2006. 56

[32] J.Y. Chen and S. Lonardi. *Biological Data Mining*. Chapman & Hall, 2009. 47, 60

[33] Z. Chen and A. Cichocki. Nonnegative Matrix Factorization with temporal smoothness and/or spatial decorrelation constraints. Technical report, Laboratory for Advanced Brain Signal Processing, RIKEN, 2005. 53

[34] A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. John Wiley & Sons, Inc., 2002. 48

[35] A. Cichocki and R. Zdunek. NMFLAB MATLAB toolbox for Non-negative Matrix Factorization. 53

[36] A. Cichocki, R. Zdunek, and S.-I. Amari. Csiszár's divergences for Non-negative Matrix Factorization: Family of new algorithms. In *LNCS*, pages 32–39. Springer, 2006. 52

[37] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using Hamming norms (how to zero in). *IEEE Trans. Knowl. Data Eng.*, 15:529–540, 2003. 16

[38] M.C. Costanzo, M.E. Crawford, J.E. Hirschman, J.E. Kranz, P. Olsen, L.S. Robertson, M.S. Skrzypek, B.R. Braun, K.L. Hopkins, P. Kondu, C. Lengieza, J.E. Lew-Smith, M. Tillberg, and J.I. Garrels. YPDTM, PombePDTM and WormPDTM: model organism volumes of the BioKnowledgeTM Library, an integrated resource for protein information. *Nucl. Acids Res.*, 29:75–79, 2001. 56

[39] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22:60–65, 2003. 16

[40] S. Datta and S. Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19:459–466, 2003. 25

[41] K. Devarajan. Nonnegative Matrix Factorization: An Analytical and Interpretive Tool in Computational Biology. *PLoS Comput. Biol.*, 4:e1000029, 2008. 47, 55, 56

[42] P. D'haeseleer. How does gene expression cluster work? *Nature Biotechnology*, 23:1499–1501, 2006. 1, 4

[43] I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001. 55

[44] I.S. Dhillon and S. Sra. Generalized Nonnegative Matrix approximations with Bregman divergences. In *Neural Information Proceedings Systems*, pages 283–290, 2005. 52

[45] V. Di Gesú, R. Giancarlo, G. Lo Bosco, A. Raimondi, and D. Scaturro. Genclust: A genetic algorithm for clustering gene expression data. *BMC Bioinformatics*, 6:289, 2005. 59

[46] D. Donoho and V. Stodden. When does Non-Negative Matrix Factorization give correct decomposition into parts? In *Seventeenth Annual Conferfence on Neural Information Processing Systems*, 2003. 54

[47] E.R. Dougherty, I. Shmulevich, L. Chen, and Z.J. Wang. *Genomic Signal Processing and Statistics*, volume 2. Hindawi Publishing Corporation, 2005. 1

[48] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3, 2002. 1, 11, 12, 15, 22, 41, 59, 61, 99

[49] S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003. 2, 13, 28, 31

[50] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, London, 1993. 14

[51] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of The National Academy of Science USA*, 95:14863–14868, 1998. 60

[52] B. Everitt. *Cluster Analysis*. Edward Arnold, London, 1993. 6, 73

[53] P. Fogel, S.S. Young, D.M. Hawkins, and N. Ledirac. Inferential, robust Non-negative Matrix Factorization analysis of microarray data. *Bioinformatics*, 23:44–49, 2007. 56

[54] E.B. Fowlkes and C.L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553–584, 1983. 19

[55] G. Frahling and C. Sohler. A fast K-means implementation using coresets. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 135–143, New York, NY, USA, 2006. ACM. 2

[56] Y. Gao and G. Church. Improving molecular cancer class discovery through sparse Non-negative Matrix Factorization. *Bioinformatics*, 21:3970–3975, 2005. 56

[57] M.R. Garey and D.S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. 8

[58] R. Gaujoux and C. Seoighe. A flexible R package for Nonnegative Matrix Factorization. *BMC Bioinformatics*, 11:367, 2010. 48, 53

[59] R.C. Gentleman, V.J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A.J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J.Y. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5:R80+, 2004. 53

[60] R. Giancarlo, G. Lo Bosco, and L. Pinello. Distance Functions, Clustering Algorithms and Microarray Data Analysis. In *Lecture Notes in Computer Science*, volume 6073, 2010. 64

[61] R. Giancarlo, G. Lo Bosco, P. Pinello, and F. Utro. The Three Steps of Clustering in the Post-Genomic Era. *Lecture Notes in Bioinformatics*, To Appear. 64

[62] R. Giancarlo, D. Scaturro, and F. Utro. Computational cluster validation for microarray data analysis: experimental assessment of Clest, Consensus Clustering, Figure of Merit, Gap Statistics and Model Explorer. *BMC Bioinformatics*, 9:462, 2008. 12, 28, 37, 60, 74, 99, 100, 108

[63] R. Giancarlo and F. Utro. Speeding up the Consensus Clustering methodology for microarray data analysis. Submitted, 2010. 15

[64] G.H. Golub and C.F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996. 48

[65] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeeck, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(531):531–537, 5439 1999. 1, 60

[66] E.F. Gonzales and Y. Zhang. Accelerating the Lee-Seung algorithm for Non-negative Matrix Factorization. Technical report, Dept. Comput. Appl. Math., Rice University, Houston, TX, 2005. 50

[67] A.D. Gordon. Clustering algorithms and cluster validation. In P. Dirschedl and R. Ostermann, editors, *Computational Statistics*, pages 503–518. Physica-Verlag, Heidelberg, Germany, 1994. 2, 13

[68] A.D. Gordon. Null models in cluster validation. In *From Data to Knowledge: Theoretical and Practical Aspects of Classification*, pages 32–44. Springer Verlag, 1996. 2, 11, 12, 13

[69] D. Guillamet, M. Bressan, and J. Vitrí. A weighted Non-Negative Matrix Factorization for local representations. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:942, 2001. 53, 54

[70] D. Guillamet, B. Schiele, and J. Vitriá. Analyzing Non-Negative Matrix Factorization for image classification. *Pattern Recognition, International Conference on*, 2:20116, 2002. 47, 54

[71] D. Guillamet, J. Vitrià, and B. Schiele. Introducing a weighted Non-negative Matrix Factorization for image classification. *Pattern Recognition Letters*, 24:2447–2454, 2003. 54

[72] D. Guillamet and M. Vitriá. Classifying faces with Nonnegative Matrix Factorization. In *Proceedings of the 5th Catalan Conference for Artificial Intelligence*, 2002. 54

[73] J. Handl, J. Knowles, and D.B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005. 2, 4, 11, 20, 27, 28, 59, 60, 73

[74] M.H. Hansen, W.N. Hurwitz, and W.G. Madow. *Sample Survey Methods and Theory Methods and Applications*, volume 1. Wiley, 1993. 15

[75] P. Hansen and P. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215, 1997. 8, 73

[76] C.W. Jr Harper. Groupings by locality in community ecology and paleoecology: tests of significance. *Lethaia*, 11:251–257, 1978. 12

[77] J.A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975. 6, 73

[78] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering of cDNAs for gene expression analysis using short oligonucleotide fingerprints. *Genomics*, 66:249–256, 2000. 61

[79] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2003. 2, 21, 47, 73

[80] A. Heger and L. Holm. Sensitive pattern discovery with "fuzzy" alignments of distantly related proteins. *Bioinformatics*, 19:i130–i137, 2003. 56

[81] L. Hood and D. Galas. The digital code of DNA. *Nature*, 421:444–448, 2003. 1

[82] P.O. Hoyer. Nonnegative sparse coding. In *IEEE Workshop on Neural Net- works for Signal Processing*, 2002. 53

[83] P.O. Hoyer and P. Dayan. Non-negative Matrix Factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004. 53

[84] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. 18

[85] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. 16

[86] M.S. Isakoff, C.G. Sansam, P. Tamayo, A. Subramanian, J.A. Evans, C.M. Fillmore, X. Wang, J.A. Biegel, S.L. Pomeroy, J.P. Mesirov, and C.W.M. Roberts. Inactivation of the Snf5 tumor suppressor stimulates cell cycle progression and cooperates with p53 loss in oncogenic transformation. *Proceedings of the National Academy of Sciences of the United States of America*, 102:17745–17750, 2005. 56

[87] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, 1988. 1, 2, 5, 6, 8, 10, 11, 14, 15, 27, 28, 39, 63

[88] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. 73

[89] W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.*, 26:189–206, 1984. 16

[90] W.B. Johnson and A. Naor. The Johnson-Lindenstrauss lemma almost characterizes Hilbert space, but not quite. In *SODA*, pages 885–891, 2009. 16

[91] I. Jung, J. Lee, S.-Y.Lee, and D. Kim. Application of Nonnegative Matrix Factorization to improve profile-profile alignment features for fold recognition and remote homolog detection. *BMC Bioinformatics*, 9:298, 2008. 56

[92] A.V. Kapp and R. Tibshirani. Are clusters found in one dataset present in another dataset ? *Biostatistics*, 8:9–31, 2007. 74

[93] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990. 2, 5, 8, 73

[94] B.M. Kelm, B.H. Menze, C.M. Zechmann, K.T. Baudendistel, and F.A. Hamprecht. Automated estimation of tumor probability in prostate magnetic resonance spectroscopic imaging: Pattern recognition vs quantification. *Magnetic Resonance in Medicine*, 57:150–159, 2007. 56

[95] M.K. Kerr and G.A. Churchill. Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. *PNAS*, 98:8961–8965, 2000. 2, 15

[96] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita. Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics*, 19:643–650, 2003. 1

[97] H. Kim and H. Park. Sparse Non-negative Matrix Factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23:1495–1502, 2007. 56

[98] P.M. Kim and B. Tidor. Subsystem Identification Through Dimensionality Reduction of Large-Scale Gene Expression Data. *Genome Research*, 13:1706–1718, 2003. 56

[99] S. Klie, Z. Nikoloski, and J. Selbig. Biological cluster evaluation for gene function prediction. *Journal of Computational Biology*, 17:1–18, 2010. 89

[100] J. Kraus and H. Kestler. A highly efficient multi-core algorithm for clustering extremely large datasets. *BMC Bioinformatics*, 11, 2010. 2

[101] W. Krzanowski and Y. Lai. A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics*, 44:23–34, 1985. 21

[102] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951. 52

[103] A. Langville, C. Meyer, R. Albright, J. Cox, and D. Duling. Algorithms, initializations, and convergence for the Nonnegative matrix factorization. In *Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006. 49

[104] J. Lawrence, A. Ben-Artzi, C. De Coro, W. Matusik, H. Pfister, R. Ramamoorthi, and S. Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics*, 25:735–745, 2006. 54

[105] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient BRDF importance sampling using a factored representation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 496–505. ACM, 2004. 54

[106] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems (Classics in Applied Mathematics)*. Society for Industrial Mathematics, new edition edition, 1987. 49

[107] D.D. Lee and H.S. Seung. Learning the parts of objects by Non-negative Matrix Factorization. *Nature*, 401:788–791, 1999. x, 47, 48, 49, 54, 55

[108] D.D. Lee and H.S. Seung. Algorithms for Non-negative Matrix Factorization. In *NIPS*, pages 556–562, 2000. x, 49, 50, 51, 52

[109] M-Y. Leung, G.M. Marsch, and T.P. Speed. Over and underrepresentation of short DNA words in Herphesvirus genomes. *Journal of Computational Biology*, 3:345–360, 1996. 8

[110] E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13:2573–2593, 2001. 40

[111] C.-J. Lin. On the convergence of multiplicative update algorithms for Non-negative Matrix Factorization. *IEEE Transactions on Neural Networks*, 18:1589–1596, 2007. 49, 50

[112] C.-J. Lin. Projected gradient methods for Non-negative Matrix Factorization. *Neural Computation*, 19:2756–2779, 2007. 49, 50, 51

[113] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967. 8

[114] F.H.C. Marriot. Practical problems in a method of cluster analysis. *Biometrics*, 27:501–514, 1971. 21

[115] H.H. Mcadams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269(5224):650–6, 1995. 1

[116] G.J. McLachlan and N. Khan. On a resampling approach for tests on the number of clusters with mixture model-based clustering of tissue samples. *J. Multivar. Anal.*, 90(1):90–105, 2004. 74

[117] L.M. McShane, M.D. Radmacher, B. Freidlin, R. Yu, M.-C. Li, and R. Simon. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. *Bioinformatics*, 18:1462–1469, 2002. 15

[118] T. Mehta, M. Tanik, and D.B. Allison. Towards sound epistemological foundations of statistical methods for high-dimensional biology. *Nature genetics*, 36:943–947, 2004. 1

[119] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985. 2

[120] G.W. Milligan and M.C. Cooper. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, 21:441–458, 1986. 19

[121] B. Mirkin. *Mathematical Classification and Clustering.* Kluwer Academic Publisher, 1996. 73

[122] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2003. x, 14, 15, 37, 59, 60, 61, 63

[123] F. Å. Nielsen, D. Balslev, and L. K. Hansen. Mining the posterior cingulate: Segregation between memory and pain components. *NeuroImage*, 27:520–532, 2005. 47

[124] O. Okun and H. Priisalu. Fast Nonnegative Matrix Factorization and its application for protein fold recognition. *EURASIP J. Appl. Signal Process*, 2006:62–62, 2006. 56

[125] P. Paatero. Least squares formulation of robust Non-negative Factor Analysis. *Chemometrics and Intelligent Laboratory Systems*, 37, 1997. 48

[126] P. Paatero. A weighted Non-negative Least Squares algorithm for three-way 'PARAFAC' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38:223–242, 1997. 48

[127] P. Paatero. The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *Journal of Computational and Graphical Statistics*, 8:854–888, 1999. 48, 52

[128] P. Paatero and U. Tapper. Positive Matrix Factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994. 48, 49

[129] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982. 42

[130] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R.D. Pascual-Marqui. Non-smooth Non-Negative Matrix Factorization (nsNMF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:403–415, 2006. 56

[131] A. Pascual-Montano, P. Carmona-Saez, M. Chagoyen, F. Tirado, J.M. Carazo, and R.D. Pascual-Marqui. bionmf: a versatile tool for Non-negative Matrix Factorization in biology. *BMC Bioinformatics*, 7:366, 2006. 56

[132] A. Pascual-Montano, F. Tirado, P. Carmona-Saez, J.M. Carazo, and R.D. Pascual-Marqui. Two-way clustering of gene expression profiles by Sparse Matrix Factorization. In *CSBW '05: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference - Workshops*, pages 103–104. IEEE Computer Society, 2005. 56

[133] V.P. Pauca, J. Piper, and R.J. Plemmons. Nonnegative Matrix Factorization for spectral data analysis. *Linear Algebra and its Applications*, 416:29–47, 2006. 53

[134] V.P. Pauca, F. Shahnaz, M.W. Berry, and R.J. Plemmons. Text mining using Non-Negative Matrix Factorizations. In *SDM*, 2004. 55

[135] C.M. Perou, S.S. Jeffrey, M. van de Rijn, C.A. Rees, M.B. Eisen, D.T. Ross, A. Pergamenschikov, C.F. Williams, S.X. Zhu, J.C.F. Lee, D. Lashkari, D. Shalon, P.O. Brown, and D. Botstein. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proceedings of the National Academy of Sciences of the United States of America*, 96:9212–9217, 1999. 1

[136] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alché−Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19:ii138–ii148, 2003. 1

[137] J. Piper, V.P. Pauca, R.J. Plemmons, and M. Giffin. Object characterization from spectral data using Nonnegative Factorization and Information Theory. In *Proceedings Amos Technical Conf.*, 2004. 49, 53

[138] J.R. Pollack, C.M. Perou, A.A. Alizadeh, M.B. Eisen, A. Pergamenschikov amd C.F. Williams, S.S. Jeffrey, D. Botstein, and P.O. Brown. Genome-wide analysis of DNA copy-number changes using cDNA microarrays. *Nature Genetics*, 23:41–46, 1999. 1

[139] I. Priness, O. Maimon, and I. Ben-Gal. Evaluation of gene-expression clustering via Mutual Information distance measure. *BMC Bioinformatics*, 8:111, 2007. 5, 60, 64

[140] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjeen, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M. Loda, E.S. Lander, and T.R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98:15149–15154, 2001. 61

[141] M.F. Ramoni, P. Sebastiani, and I.S. Kohane. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, 99:9121–9126, 2002. 3, 124

[142] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971. 18

[143] Y. Raviv and N. Intrator. Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8:355–372, 1996. 15

[144] J.A. Rice. *Mathematical Statistics and Data Analysis.* Wadsworth, 1996. 47, 73

[145] C. Van Rijsbergen. *Information Retrieval, second edition.* Butterworths, London, 1979. 19

[146] D.T. Ross, U. Scherf, M.B. Eisen, C.M. Perou, P. Spellman, V. Iyer, S.S. Jeffrey, M. van de Rijn, M. Walthama, A. Pergamenschikov, J.C.F. Lee, D. Lashkari, D. Shalon, T.G. Myers, J.N. Weistein, D. Botstein, and P.O. Brown. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, 24:227–235, 2000. 1

[147] V. Roth, T. Lange, M. Braun, and J. Buhmann. A resampling approach to cluster validation. In *Proceedings 15th Symposium in Computational Statistics*, pages 123–128, 2002. 42

[148] K. Sachs, D. Gifford, T. Jaakkola, P. Sorger, and D.A. Lauffenburger. Bayesian network approach to cell signaling pathway modeling. *Science's STKE : signal transduction knowledge environment*, 2002, 2002. 1

[149] W.S. Sarle. Cubic clustering criterion. Technical report, SAS, 1983. 11, 12

[150] S. Seal, S. Comarina, and S. Aluru. An optimal hierarchical clustering algorithm for gene expression data. *Information Processing Letters*, 93:143–147, 2004. 2

[151] F. Shahnaz, M.W. Berry, V.P. Pauca, and R.J. Plemmons. Document clustering using Nonnegative Matrix Factorization. *Information Process. Manage.*, 42:373–386, 2006. 55

[152] R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In T. Jiang, T. Smith, Y. Xu, and M. Q. Zhang, editors, *Current Topics in Computational Biology*, pages 120–161. MIT Press, Cambridge, Ma., 2003. 5, 61

[153] I. Shmulevich, E.R. Dougherty, S. Kim, and W. Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18:261–274, 2002. 1

[154] A. Silvescu and V. Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13:61–78, 2001. 1

[155] P. Smolen, D.A. Baxter, and J.H. Byrne. Modeling transcriptional control in gene networks–methods, recent results, and future directions. *Bulletin of mathematical biology*, 62:247–292, 2000. 1

[156] M. Smolkin and D. Ghosh. Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4, 2003. 15

[157] J.A. Snyman. Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms. *Structural and Multidisciplinary Optimization*, 31:249, 2006. 49

[158] T.P. Speed. *Statistical analysis of gene expression microarray data*. Chapman & Hall/CRC, 2003. 5, 47

[159] P.T. Spellman, G. Sherlock, M.Q. Zhang, V. R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle regulated genes of the yeast Saccharomyces Cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998. 61

[160] R.E. Strauss. Statistical significance of species clusters in association analysis. *Ecology*, 63:634–639, 1978. 12

[161] A.I. Su, M.P. Cooke, K.A. Ching, Y. Hakak, J.R. Walker, T. Wiltshire, A.P. Orth, R.G. Vega, L.M. Sapinoso, A. Moqrich, A. Patapoutian, G.M. Hampton, P.G. Schultz, and J.B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *Proceedings of the National Academy of Sciences of the United States of America*, 99:4465–4470, 2002. 61

[162] D. Swagatam, D. Sambarta, B. Arijit, A. Ajith, and K. Amit. On stability of the chemotactic dynamics in bacterial-foraging optimization algorithm. *Trans. Sys. Man Cyber. Part A*, 39:670–679, 2009. 49

[163] P. Tamayo, D. Scanfeld, B.L. Ebert, M.A. Gillette, C.W.M. Roberts, and J.P. Mesirov. Metagene projection for cross-platform, cross-species characterization of global transcriptional states. *Proceedings of the National Academy of Sciences of the United States of America*, 104:5959–5964, 2007. 56

[164] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistics. *Journal Royal Statistical Society B.*, 2:411–423, 2001. 11, 12, 22, 25

[165] L.N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM: Society for Industrial and Applied Mathematics, 1997. 48

[166] J.A. Tropp. Literature survey: Non-Negative Matrix Factorization. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.9645. 49

[167] G. Valentini. Mosclust: a software library for discovering significant structures in bio-molecular data. *Bioinformatics*, 23:387–389, 2007. 28

[168] A. Vassiliou, L. Ignatiades, and M. Karydis. Clustering of transect phytoplankton collections with a quick randomization algorithm. *Journal of experimental marine biology and ecology*, 130:135–145, 1989. 12

[169] G. Wang, A.V. Kossenkov, and M.F. Ochs. LS-NMF: A modified non-negative matrix factorization algorithm utilizing uncertainty estimates. *BMC Bioinformatics*, 7:175, 2006. 53

[170] K. Wang, N. Zheng, and W. Liu. Natural image matting with Non-negative Matrix Factorization. In *ICIP (2)*, pages 1186–1189, 2005. 54

[171] Y. Wang, Y. Jia, C. Hu, and M. Turk. Fisher Non-negative Matrix Factorization for learning local features. In *Asian Conference on Computer Vision*, 2004. 52

[172] X. Wen, S. Fuhrman, G.S. Michaels, G.S. Carr, D.B. Smith, J.L. Barker, and R. Somogyi. Large scale temporal gene expression mapping of central nervous system development. *Proceedings of The National Academy of Science USA*, 95:334–339, 1998. 60

[173] S. Wild. *Seeding Non-negative Matrix Factorizations with the spherical K-Means clustering.* PhD thesis, University of Colorado, 2003. 49

[174] I.H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Academic Press, San Diego, CA,, 2000. 47

[175] R.D. Wolfinger, G. Gibson, E.D. Wolfinger, L. Bennet, H. Hamadeh, C.A. Bushel, and R.S. Paules. Assessing gene significance from cDNA microarray expression data via mixed models. *Journal of Computational Biology*, pages 625–637, 2001. 15

[176] W. Xu, X. Liu, and Y. Gong. Document clustering based on Non-negative Matrix Factorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003. 55

[177] J. Yamagishi, H. Kawai, and T. Kobayashi. Phone duration modeling using gradient tree boosting. *Speech Commun.*, 50:405–415, 2008. 49

[178] M. Yan and K. Ye. Determining the number of clusters with the weighted Gap Statistics. *Biometrics*, 63:1031–1037, 2007. 25, 79

[179] E.-J. Yeoh, M.E. Ross, S.A. Shurtleff, W.K. Williams, D. Patel, R. Mahfouz, F.G. Behm, S.C. Raimondi, M.V. Relling, A. Patel, C. Cheng, D. Campana, D. Wilkins, X. Zhou, J. Li, H. Liu, C.-H. Pui, W.E. Evans, C. Naeve, L. Wong, and J.R. Downing. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002. 61

[180] K.Y. Yeung. *Cluster Analysis of Gene Expression Data*. PhD thesis, University of Washington, 2001. 19

[181] K.Y. Yeung, D.R. Haynor, and W.L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17:309–318, 2001. 25, 64