FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Meta-optimizations for Cluster Analysis

by

Tomáš Bartoň

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Applied Mathematics

Prague, July 2019

Supervisor:
doc. Ing. Pavel Kordík, Ph.D.
Department of Applied Mathematics
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

This dissertation thesis deals with advances in automation of cluster analysis, that belongs to a group of unsupervised methods in Machine Learning field. Over more that 60 years numerous methods have been developed and successfully applied in many areas that deals with data processing. As data analysis area is attracting more users, the necessity of organizing clustering algorithms and its results becomes more important. Moreover, each clustering algorithm is controlled by set of hyper-parameters that are difficult to optimize, because we cannot use external criteria as in case of supervised methods.

In this thesis, we investigate how unsupervised clustering criteria can help us to select most appropriate clustering algorithm or ensemble. We also work towards AutoML in clustering which is much more challenging than in other machine learning domains.

In particular, main contributions of the dissertation thesis are as follows:

1. Chameleon 2: a multi-objective hierarchical algorithm.
2. A visualization method for comparing the rankings of two evaluation measures.
3. Multi-objective ranking as a Pareto optimization.
4. A framework for automatic meta learning clustering based on meta features to describe the dataset.

Keywords:
clustering, cluster analysis, clustering validation, unsupervised learning, clustering evaluation, meta-optimization, cluster ensembles.

Acknowledgements

I would like to thank to my supervisor doc. Ing. Pavel Kordík, Ph.D. for his patience and guidance, to prof. Ing. RNDr. Martin Holeňa, CSc. for advises in my research and corrections of many drafts. To Ing. Tomáš Brůna belongs huge thanks for enthusiasm, hard-work and constant optimism. I am grateful to Prof. George Karypis for motivation to continue in my research. Furthermore I would like to thank to Prof. Jan Vitek for kinds words and support in my work and to Dr. Arthur Zimek for detailed review and tolerance to many mistakes committed.

Finally, I would like to thank my wife Marcela for the endless patience she had in bearing with me for all these years.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

Common notation

N	Total number of data points (instances) in the dataset
D	Dataset dimensionality (size of observation vector)
$\mathbb{C}$	A set of K clusters, commonly refered as a clustering
K	Total number of clusters in clustering $\mathbb{C}$, same as $\lvert\mathbb{C}\rvert$
C_k	k-th cluster in clustering $\mathbb{C}$
$\mathbb{N}$	Natural numbers set
n_k	size of k-th cluster
$\boldsymbol{c}_k$	Centroid of cluster C_k
X	Input dataset containing N vectors of size D
$\boldsymbol{x}_k$	Data vector of size D
N_t	Total number of pairs of data points in the dataset
N_w	Total number of pairs of data points belonging to the same cluster
N_b	Total number of pairs of observations belonging to different clusters

Common Mathematical Functions and Operators

$\mathbf{x}$	Vector $\mathbf{x}$
b_i	the i^{th} element of vector $\mathbf{b}$
$\|\|\mathbf{b}\|\|$	Norm of vector $\mathbf{b}$
$\dim \mathbf{b}$	Dimension of vector $\mathbf{b}$
$\mathbf{A}$	Matrix $\mathbf{A}$
$a_{i,j}$	Element of matrix $\mathbf{A}$ at the i^{th} row, and the j^{th} column
$\mathbf{A}^{-1}$	Inverse matrix to matrix $\mathbf{A}$
$\mathbf{A}^{\intercal}$	Transposed matrix to matrix $\mathbf{A}$
$\|\|\mathbf{A}\|\|$	Norm of matrix $\mathbf{A}$
$\mathrm{Tr}(\mathbf{A})$	Trace of an n-by-n square matrix $\mathbf{A}$
$\operatorname{rank} \mathbf{A}$	Rank of matrix $\mathbf{A}$ — how many independent rows/columns it has
$\max\{a, b\}$	Maximum of a and b, a when $a \geq b$, b when $a < b$
$\min\{a, b\}$	Minimum of a and b, a when $a \leq b$, b when $a > b$
$O(x)$	The big O notation
$\Theta(x)$	The big Θ notation

Basic Definitions

The **standard deviation** σ_x is computed as:

$$\sigma_x = \sqrt{\frac{1}{n}\left((x_1 - \overline{x})^2 + (x_2 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2\right)} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2}$$

where $\overline{x}$ is *arithmetic mean* or *average* defined as[1]:

$$\overline{x} = \frac{1}{n}\left(x_1 + x_2 + \cdots + x_n\right) = \frac{1}{n}\sum_{k=1}^{n} x_k$$

The **mean absolute deviation**, sometimes referred to also as *mean absolute error*:

$$\sigma_x^A = \frac{1}{n}(|x_1 - \overline{x}| + |x_2 - \overline{x}| + \cdots + |x_n - \overline{x}|) = \frac{1}{n}\sum_{k=1}^{n}(|x_k - \overline{x}|)$$

Miscellaneous Abbreviations

AIC	Akaike Information Criterion
ARFF	Attribute-Relation File Format
BIC	Bayesian Information Criterion
NMI	Normalized Mutual Information
NSGA	Non-dominated Sorting Genetic Algorithm
PCA	Principal Component Analysis
SOM	Self-Organizing Map

[1] Different authors use various notation, this definitions are just for making clear notation used in here.

Chapter 1

Introduction

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."

— Lewis Carroll, *Alice in Wonderland*

1.1 Motivation

The goal of this dissertation thesis is to introduce a methodology for a systematic exploration of clustering space and to propose an automated unsupervised approach for data clustering. Nowadays anyone who tries to cluster data, whether a data-mining expert or a common user, is faced with an unclear decision over which algorithm to use and how to configure its (many) parameters. It is extremely difficult for users to decide which algorithm would be the best choice for a given set of data [2]. The following chapters set out guidelines that should make clustering algorithms easier to use and to evaluate. Currently, many data scientists choose a particular algorithm rather for its speed or thanks to their previous experience with that algorithm on a completely different problem. Cluster analysis is typically employed in the initial phase of exploring raw data, where prior knowledge is minimal. Having automated methods is crucial, especially in the the modern era of "big data" where manual data investigation would be overwhelming.

1.2 Problem Statement

Clustering methods have been developed since the 1960s, yet there is still no universal method or methodology for selecting an appropriate clustering algorithm for given data. With the growing amounts of data available online and more being generated every second comes higher demand for fast and scalable clustering algorithms. Exploring full state space with all algorithms available and its meaningful configurations manually is simply not feasible with larger datasets.

In the machine learning field there is only a limited amount of labeled data, but almost an unlimited supply of unlabeled data. Simplifying the clustering task may help to solve other related problems, such as pattern classification and rule extraction from data.

1.3 Previous Results

This work follows on from my Master's Thesis topic which was "Cluster analysis of cell profile responses" and dealt with processing time-series data for the Institute of Molecular Genetics of the Czech Academy of Sciences[1]. I participated in the CZ-OPENSCREEN project where the methods described here are tested on real data. This project dealt with High-Throughput Screening where methods for fast processing of large datasets are needed.

1.4 Contributions of the Dissertation Thesis

Following the original goals proposed in [3] the main contributions can be summarized as:

1. Chameleon 2: a multi-objective hierarchical algorithm.
2. A visualization method for comparing the rankings of two evaluation measures.
3. Multi-objective ranking as a Pareto optimization.
4. A framework for automatic meta learning clustering based on meta features to describe the dataset.

1.5 Structure of the Dissertation thesis

The dissertation thesis is organized into 5 chapters as follows:

1. *Introduction*: Describes the motivation behind improving the clustering process.
2. *Clustering Background*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.
3. *Chameleon 2*: Briefly describes published clustering algorithm.
4. *Clustering Evaluation*: Summarizes methods for evaluating clustering results and introduces a multi-objective ranking.
5. *Clustering Ensembles*: Describes commonly used ensemble techniques in the clustering context.

[1] `http://www.img.cas.cz/`

6. *AutoML Clustering*: Describes proposed meta-learning framework that builds up on previous chapters.

7. *Conclusions*: Summarizes possibilities of further research.

Chapter 2

Clustering Background

Clustering techniques partition objects into groups of *clusters* so that objects within a cluster are *similar* to one another and *dissimilar* to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based on a *distance function* [4].

In real-world application clustering can be misled by the assumption of uniqueness of the solution. Because of noise, intrinsic ambiguity in data and optimization models attempting to maximize a fitness function [5]. Most clustering algorithm search for one optimal solution based on a pre-specified clustering criterion [6].

According to Pang-Ning Tan [7] clustering can be used for *utility* or *understanding*. Clustering for utility is used when we need to get an abstraction from group of individual objects. By understanding is meant a discovery of data groups that share common characteristics. However clustering might be applied during an early stage of data exploration, before users know the data well enough to define a suitable clustering criteria. Resulting clustering could be negatively influenced by user's (strong) assumption about data. This creates a chicken-or-the-egg problem where knowing how to define a good clustering criterion requires understanding the data. And on the other hand clustering is used to understand the data [6].

Grouping items is a very elemental task that found applications in many areas including many scientific areas like astronomy, artificial intelligence, biology [8], chemistry, customer relationship management [9], history, information retrieval [10], image processing [11], pattern recognition, psychology [12] or recommender systems [13]. Clustering has also been successfully applied in many real-world scenarios in areas like geography, marketing [14] or medicine. Although the terminology and data processing pipelines differs, the base algorithms itself have wide range of applications.

2.1 Definition of a cluster

As it was mentioned above, there is no universal definition for the term *cluster*. Here we have some definitions that are commonly used [15]:

1. **Well-separated cluster**: A cluster is a set of objects such, that any object in a cluster is closer (or more similar) to every other object in the cluster, than to any object not in the cluster.
2. **Center-based cluster**: A cluster is a set of objects such, that an object in a cluster is closer (or more similar) to the *center* of a cluster, than to the center in any other cluster. The center of a cluster is often a *centroid*, the average of all the objects in the cluster, or a *medoid*, the most "representative" object of a cluster.
3. **Contiguous cluster**: A cluster is a set of objects such, that a point in a cluster is closer (or more similar) to one or more other objects in the cluster, than to any object not in the cluster.
4. **Density-based cluster**: A cluster is a dense region of points which is separated by low-density regions, from other regions of high density. This definition is more often used when the clusters are irregular or intertwined, and when the noise of outliers is present.
5. **Similarity-based cluster**: A cluster is a set of objects that are *similar* and objects in other clusters are *dissimilar*. A variation of this is to define a cluster as a set of objects, that together create a region with uniform local property, e.g. density or shape.

2.2 Preprocessing

Changing units or scale may lead to different ranges for each parameter and thus it might change the whole clustering structure. In some cases a user might want to put higher significance on a specific parameter, but generally parameter ranges should be standardized. Standardizing will give all parameters an equal weight [4].

Now we can apply standardization to our data matrix. In what follows $\mathbf{x}_i$ is a vector with D values, representing a sample or a i^{th} row of the data matrix. x_{ij} is a value of j^{th} parameter of the i^{th} sample and ${x_{ij}}'$ is a standardized value.

For standardization we can use the following:

1. **Maximum value** – Divide each attribute value of an object by the maximum value of that attribute. This will put all values at an interval between -1 and 1.

$$ {x_{ij}}' = \frac{x_{ij}}{\max |\mathbf{x}_j|} \tag{2.1} $$

2. **Z-score normalization** – For each value subtract the mean $\overline{x}_j$ of that parameter and divide by its standard deviation.

$$ {x_{ij}}' = \frac{x_{ij} - \overline{\mathbf{x}}_j}{\sigma_j} \tag{2.2} $$

If data are normally distributed, then most attribute values will lie between -1 and 1.

3. **Standardized measurement** – For each value subtract the mean $\overline{x}_j$ of that parameter and divide by parameter's mean absolute deviation:

$$x_{ij}{}' = \frac{x_{ij} - \overline{\mathbf{x}}_j}{\sigma_j^A} \tag{2.3}$$

4. **Min-max normalization** – Performs a linear transformation of values into a range between r_{min} and r_{max}. This could be helpful when we want to get rid of negative values or simply unify ranges of values to whatever interval that suits our needs [4].

$$x_{ij}{}' = (x_{ij} - \min|\mathbf{x}_j|) * \frac{r_{max} - r_{min}}{\max|\mathbf{x}_j| - \min|x_j|} + r_{min} \tag{2.4}$$

This is basically a generalized version of Maximum value standardization.

All of them are unit-less because both the numerator and the denominator are expressed in the same units. The first approach might not produce good results unless the parameters are uniformly distributed, but both first and second approaches are sensitive to outliers. The third approach is the most robust in the presence of outliers, although elimination of outliers should be done in the data cleaning phase, if it is possible [15].

The preceding description might convey the impression that standardization is necessary for all data, however it is just one possibility data preprocessing. If input data are all in the same units, or they have an absolute meaning, they should not be standardized [16].

2.3 Measuring distance

Measuring distance (or more generally proximity) is one of the key parameters of a clustering method, many users might know names of these metrics but it is important to realize also their advantages and disadvantages and use them wisely.

There are many ways to measure quantitative variables, the most common is **Euclidean distance**. Suppose we have two vectors $\mathbf{x} = (x_1, \ldots, x_D)$ and $\mathbf{y} = (y_1, \ldots, y_D)$ in an Euclidean D-dimensional space.

Euclidean distance Euclidean distance is a function $p : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ that assigns to any two vectors in Euclidean D-space the number:

$$p(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{D} (x_j - y_j)^2} \tag{2.5}$$

and so gives the "standard" distance between any two vectors in $\mathbb{R}^D$[17].

Cosine distance Cosine distance sometimes also called *coefficient of correlation*, represents an angle between two vectors. The angle is computed as a product of two vectors divided by the product of their length:

$$cos\phi = \frac{\mathbf{x}\ \mathbf{y}}{|\mathbf{x}|\ |\mathbf{y}|} \tag{2.6}$$

The length of a vector is the root of square of its coordinates:

$$|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + \cdots + x_D^2} \tag{2.7}$$

Putting all together we get:

$$cos\phi = \frac{x_1\, y_1 + x_2\, y_2 + \cdots + x_n\, y_n}{\sqrt{x_1^2 + x_2^2 + \cdots + x_D^2}\ \sqrt{y_1^2 + y_2^2 + \cdots + y_D^2}} \tag{2.8}$$

When we change the notation of vectors into coordinates, we get:

$$cos\phi = \frac{\sum_{j=1}^{D} x_j\ y_j}{\sqrt{\sum_{j=1}^{D} (x_j\ y_j)^2}} \tag{2.9}$$

While there are many other ways of measuring distance, most methods are designed to work with either Euclidean or Cosine distances. Switching between distance functions during clustering process might have undesired consequences. Thus in the following parts Euclidean distance is used, unless stated otherwise.

2.4 Algorithms

Throughout the data analysis history many clustering algorithms with different characteristics and different purposes have been proposed and investigated [18]. Most of the clustering algorithms, in particular the most traditional and popular ones, require that the number of clusters be defined either a priori or a posteriori by the user [19]. Examples are the well-known k-means [20], EM (Expectation Maximization) [21], and hierarchical clustering algorithms [1].

2.4.1 Partitioning algorithms

2.4.1.1 K-means clustering

K-means [20, 22] is probably the most simple exclusive clustering algorithm. Each object is assigned precisely to one cluster, that means an exclusive, partitioning algorithm. At

the beginning we have to decide how many clusters we want, and this value is called k; obviously k must be much smaller than the total number of objects, otherwise there is nothing to divide. We select k objects and assign each of them to a separate cluster. We can select those objects randomly, or probably the most intuitive method would be to choose objects that lie from each other as far as possible.

Then we iterate over all data points, for each of them we find the nearest cluster center (*centroid*), after assigning the data point to a cluster, centroids location must be updated. Note that the center of a cluster is usually a non-existing object. This process continues until either maximum number of iterations is reached or the criterion function converges. K-means clustering is a greedy algorithm which is guaranteed to converge to a local minimum, but the minimization is known to be NP-Hard problem [23]. For instance as a criterion function a **squared-error** could be used:

$$E = \sum_{k=1}^{K} \sum_{x \in C_k} |\mathbf{x} - \mathbf{c}_k|^2 \tag{2.10}$$

where x is the point representing the object in space, $\mathbf{c}_k$ is the mean of cluster C_k.

Algorithm 2.1 k-means

```
1: procedure KMeans(dataset)
2:     Select k objects as initial centroids
3:     while convergence criterion not reached do
4:         (re)assign each object to the cluster to which is object most similar
5:         re-calculate cluster center
6:         calculate criterion function (squared-error)
7:     end while
8: end procedure
```

This algorithm works well when clusters are compact and rather separated from one another. The method is relatively scalable and efficient in processing large amounts of data, the space requirements are basically $O(N * D)$, where N is the number of objects and D is the number of attributes. The computational complexity is $O(I * K * N * D)$, where K is the number of clusters and I is a number of iterations. Normally, $K \ll N$ and $I \ll D$ [4, 15].

The result of this algorithm mainly depends on appropriate selection of initial clusters, when we randomly select more points from one region, we will probably obtain unexpected results. The algorithm might converge to a local minimum without finding the global minimum. This depends again on initialization. Furthermore, the probability of choosing an element from each cluster is pretty low, especially when we consider that the numbers of objects in each cluster are not equal (we might have a cluster with very few objects).

The k-means method has problems when desired cluster has either different size or shape other than spherical, or it contains noise and outliers (see Figure 2.1). The main

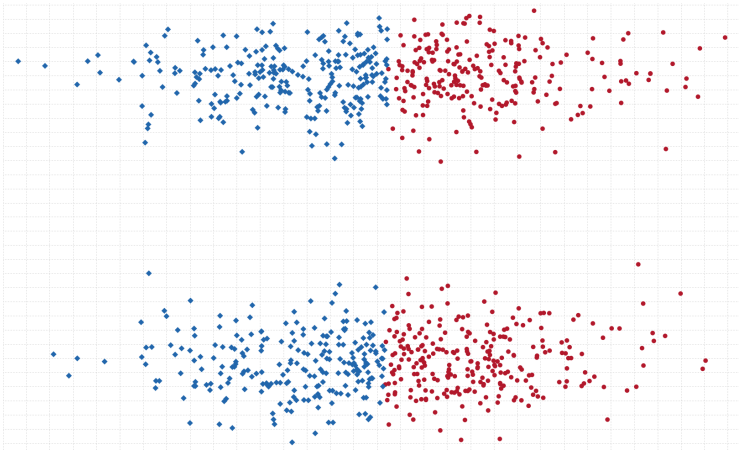


Figure 2.1: Single k-means runs on the *long1* dataset using $k = 2$. K-Means algorithm fails to reveal non-spherical clusters. Both supervised indexes Ajusted Rand Index and NMI assigns this clustering score 0.0, even though there is 50% assignment error.

disadvantage is that we have to specify the number of clusters before starting the process, so that it cannot be used for discovering new groups in unknown data. However, it is one of the fastest methods with low memory requirement.

Many extensions and attempts to improve the k-means algorithm have been proposed. K-means++ [24] attempts to carefully select the initial centroids in order to avoid ending up in a local minimum. The algorithm follows a simple probability-based approach: the first centroid is selected randomly while the next one is the farthest away choice. The centroid selection is decided based on a weighted probability score. Elkan [25] tried to accelerate k-means by avoiding distance computation to centroids that are too far away. Kanungo [26] proposed to store data points in a *kd*-tree structure, in order to speed-up computations.

2.4.1.2 K-medoids clustering

A k-medoids method tries to find a non-overlapping set of clusters such that each cluster has a most representative point, that means a point that is located in the centre with respect to some measurement (e.g. distance). These representatives are called *medoids*.

The cost of each cluster could be evaluated as follow:

$$\text{cost}_i = \sum_{j=1}^{n_i} p(\mathbf{c}_i, \mathbf{x}_j) \tag{2.11}$$

where cost_i is a cost for i^{th} cluster, $p(\mathbf{c}_i, \mathbf{x}_j)$ is a proximity between medoid in i^{th} cluster and j^{th} object in this cluster. For similarities (respective dissimilarities) we would like to have this sum as small (respective large) as possible.

Algorithm 2.2 k-medoids algorithm

```
1: procedure KMedoids(dataset)
2:     choose K objects
3:     while has new configuration? do
4:         consider replacing medoids with non-medoids objects
5:         calculate cost for each cluster
6:         select configuration with best (lowest or highest) cost
7:     end while
8:     associate non-medoids objects with closest medoids
9: end procedure
```

Finding the best medoids requires trying all possibilities, which is computationally expensive. And the same disadvantage as for k-means, we have to know the number of clusters before starting [15].

2.4.1.3 CLARANS

CLARANS [27] is a method that was developed as a mixture of k-medoid method called PAM [16] (Partitioning Around Medoids) and CLARA (Clustering LARge Applications), which repeatedly samples subsets of data and in a few iterations tries to optimize the cost of selected medoids.

CLARANS uses randomized search to improve both previously mentioned, the method is described in Algorithm 2.3 output.

Algorithm 2.3 CLARANS algorithm

```
1: procedure CLARANS(dataset)
2:     j := 0
3:     while j < 2 do
4:         randomly choose k objects as medoids
5:         while i < max(250, k * (m − k)) do
6:             randomly swap medoids for non-medoids
7:             calculate cost for each cluster
8:             if found better configuration then
9:                 apply configuration
10:            end if
11:            i := i + 1
12:        end while
13:    end while
14:    associate non-medoids objects with closest medoids
15: end procedure
```

Unlike PAM, CLARANS does not consider all possible swaps, but only a random se-

lection. It differs from CLARA in that it works with all data objects. The algorithm was meant for use in spatial data mining.

Neither CLARANS nor k-means are able to find the clusters non-spherical clusters (in 2D). However, CLARANS seems to be a suitable version of k-medoid method, which does not guarantee finding best solution, but might provide quite a good solution in reasonable time [28].

2.4.1.4 Mean Shift

Mean Shift clustering [29] is a non-parametric method that has been successfully applied in computer vision and pattern recognition. The mean shift procedure tries to determine the local maxima or modes present in the data distribution. The algorithm is based on the Parzen window kernel density estimation method. It starts with each point and then performs a gradient ascent procedure until convergence [30]. The algorithm is outlined in Algorithm 2.4. The multi-variate Parzen window kernel density estimate $f(x)$ is obtained with kernel $Q(x)$ and window radius h:

$$f(x) = \frac{1}{Nh^D} Q\left(\frac{x - x_i}{h}\right) \tag{2.12}$$

$$m_h(x) = \frac{\sum_{i=1}^{N} x_i \cdot g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} \tag{2.13}$$

Algorithm 2.4 Mean Shift algorithm

```
1: procedure MeanShift(dataset)
2:     Select K random points as the modes of the distribution
3:     while Modes not converged do
4:         For each given mode x calculate the mean shift vector m_h(x)
5:         Update the point x = m_h(x)
6:     end while
7: end procedure
```

2.4.1.5 Self-organizing maps (SOM)

Self-organizing (SOM) maps were developed by Kohonen [31] and belongs to one of the most popular neural network models. It is an unsupervised learning algorithm with a simple structure and computational form, and is motivated by the retina-cortex mapping. In a similar manner like k-means, the data points are assigned to their closest centroids.

The basic algorithm can includes following steps:

1. Initialize k cluster centers, i.e. $\overline{C} = \{\overline{\mathbf{c}}_1, \ldots, \overline{\mathbf{c}}_k\}$

2. For each data point $\mathbf{x}_i \in X$:

   a) Assign $\mathbf{x}_i$ to a cluster C_j with minimal distance between $\mathbf{x}_i$ and its cluster center $\overline{\mathbf{c}}_j$

   b) Update the cluster center $\overline{\mathbf{c}}_j$, which is the weight vector of SOM's output units:

$$\overline{\mathbf{c}}_j = \overline{\mathbf{c}}_j + h[\mathbf{x}_i - \mathbf{c}_j] \tag{2.14}$$

   where $h \in [0, 1]$ is the degree of neighbourhood. In addition to updating the center $\overline{\mathbf{c}}_j$ of the cluster that $\mathbf{x}_i$ belongs to, all the cluster centers that are in the neighborhood of $\overline{\mathbf{c}}_j$ on the grid map are also updated. This neighborhood-based propagation is controlled by h, which can be specified using the neighbourhood functions such as the bell-shaped (Gaussian-like) and the square. This process is repeated for all $\mathbf{x}_i \in X$.

Despite its innovative concept and reported success, the major disadvantage of SOM remains its high computational complexity. Many variations of SOM have been introduced in the literature, see [32, 33] as examples of more recent modifications.

2.4.2 Hierarchical Clustering

A hierarchical clustering method groups objects into trees of clusters, we can further divide them into two groups, depending whether the decomposition is done by top-down (divisive) or bottom-up (agglomerative) approach. There is one important drawback (against k-means, e.g.), once we divide (or join) two clusters this decision can not be adjusted.

In either agglomerative or divisive hierarchical clustering, one can specify the desired number of clusters as a terminating condition so that the hierarchical clustering process will terminate when the process reaches the desired number of clusters, but it is not necessary to do so, as in the case of k-means [4].

The result of clustering is often graphically displayed in a tree-like diagram called a **dendrogram**.

2.4.2.1 Divisive Hierarchical Clustering

Divisive clustering starts with all objects in one cluster, subdividing them into smaller pieces until each object has its own cluster or until it satisfies some criteria, such as a desired number of clusters or the distance between two closest clusters is below a certain threshold distance.

A simple version of a divisive approach is shown in Algorithm 2.5.

Divisive methods are not frequently used because of the difficulty of making the right decision at the top level.

Algorithm 2.5 Divisive clustering algorithm

```
1: procedure DivisiveHC(dataset)
2:     while has free objects do
3:         compute minimum spanning tree for the proximity graph
4:         create a new cluster by breaking the link to furthest object
5:     end while
6: end procedure
```

2.4.2.2 Agglomerative Hierarchical Clustering

The idea behind this algorithm is simple. We start with each object as a separate cluster, and incrementally we join the closest pair of clusters together, until we end-up with one super-cluster, which contains everything. Most hierarchical algorithms belong to this category, they differ in metric used for computing proximity.

Algorithm 2.6 Hierarchical agglomerative clustering algorithm

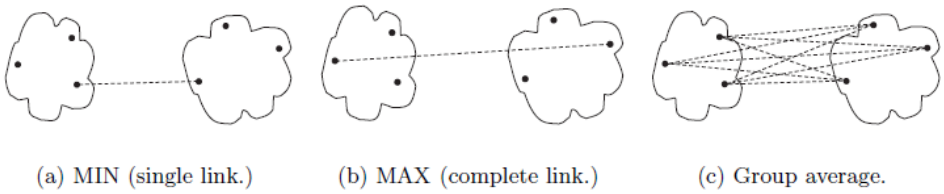
```
1: procedure DivisiveHC(dataset)
2:     assign each object to its own cluster
3:     repeat
4:         calculate (update) distance between each pair of clusters
5:         select closes pair of clusters and join them into one
6:     until all clusters are joined into single cluster
7: end procedure
```

It would be very inefficient to calculate the proximity (distance) between each pair of clusters for each pass though the loop, especially when only two objects are changed. The common approach is to create a *proximity matrix*, which records the distances between all clusters [34].

2.4.2.3 Proximity between clusters

For simplifying the computation of distances sometimes not all members of the cluster are taken into consideration. We can choose the closest members of those clusters, then it is called MIN or the furthest member, which would be MAX. However when the same approach is applied to counting similarity, the most similar objects (closest) would be called MAX and the other way round, which becomes confusing. For that reason we usually use a *single link* for closest and most similar objects, and a *complete link* for the opposite [7].

Single link The similarity of two clusters is the similarity of their **most similar** members. You start with all clusters and add links between them, with the strongest links first, then these single links combines the closest clusters. A single link is good at handling non-elliptical shapes, but it is sensitive to noise and outliers.



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Figure 2.2: Cluster proximity definitions. Image credits: [7].

Complete link The similarity of two clusters is the similarity of their **most dissimilar** members. You start with all clusters and add links between them, strongest first, until all clusters are completely linked (i.e. clusters form a *clique*[1]). A complete link is less susceptible to noise and outliers. But on the other hand, it can break large clusters and has trouble with convex shapes.

Group average Another approach is the group average, which counts the average of all pairs of points from two clusters (see 2.2). This is an intermediate approach between single and complete link.

Centroid link Yet another way is using the centroid link, which computes the distance between a single pair of two cluster's centroids.

Ward's method Ward's method [35] uses for defining proximity the increase in the squared error (see 2.10), that is computed when two clusters are merged. That is the same method as k-means uses (2.4.1.1).

The following are the widely used distance measurement, where $\mathbf{c}_i$ is the mean for cluster C_i, n_i is the number of objects in cluster C_i and $|p - p'|$ is the distance between an object in cluster C_i and another object in cluster C_j.

$$d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'| \tag{2.15}$$

$$d_{centroid}(C_i, C_j) = |\mathbf{c}_i - \mathbf{c}_j| \tag{2.16}$$

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'| \tag{2.17}$$

$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'| \tag{2.18}$$

The Lance-Williams formula for cluster proximity The proximity between clusters Q and R, where R is formed by merging clusters A and B is defined by Lance-Williams formula [36]:

[1] from graph theory, sets of elements where each pair of elements is connected

Clustering method	α_A	α_B	β	γ
single link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
complete link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
group average	$\frac{n_A}{n_A+n_B}$	$\frac{n_B}{n_A+n_B}$	0	0
centroid link	$\frac{n_A}{n_A+n_B}$	$\frac{n_B}{n_A+n_B}$	$\frac{-n_An_B}{(n_A+n_B)^2}$	0
Ward's method	$\frac{n_A+n_k}{n_A+n_B+n_k}$	$\frac{n_B+n_k}{n_A+n_B+n_k}$	$\frac{-n_k}{n_A+n_B+n_k}$	0

Table 2.1: Lance-Williams formula's parameters values for different linkage methods [1].

$$p(R,Q) = \alpha_A\, p(A,Q) + \alpha_B\, p(B,Q) + \beta\, p(A,Q) + \gamma\, |p(A,Q) - p(B,Q)| \tag{2.19}$$

In words, this formula says that if you merge clusters A and B to form cluster R, then the distance of the new cluster R to existing cluster Q is a **linear** function of distances between Q and the original clusters A and B.

As you can see in Table 2.1, in Ward's method, no matter how different it seems, the proximities values will be very similar to group average method. n_A stands for the number of elements in cluster C_A and n_k is a total number of elements.

2.4.2.4 Complexity

The storage of a proximity matrix, which is usually used by hierarchical clustering algorithms, requires storage of N^2 proximities, where N is a number of objects. When we consider just a triangular matrix it is $(N^2 - N)/2$, which is anyway is still $O(N^2)$. It would be possible to count the proximities on the fly, however it would require repeated calculations and therefore more time. Generally the hierarchical clustering requires $O(N^3)$ computational time, as the proximity matrix needs to be updated each time two clusters are joined.

Sibson [37] proposed a SLINK algorithm that is optimally efficient and implements single-linkage clustering with time complexity $O(N^2)$ and space complexity $O(N)$. The algorithm introduces a compact pointer representation of dendrogram using pointers, which allows faster computation.

Inspired by SLINK, Defays [38] described CLINK algorithm for complete-linkage with time complexity $O(N^2)$ as well.

Hierarchical clustering can not be viewed as globally optimizing an objective function. Instead, algorithms decide locally at each step which cluster should be merged or divided. This approach avoids solving hard optimization problems (which are solved in k-means), furthermore it does not have problems with local minima or difficulties with choosing initial points.

On the other hand, the space and time complexity $O(N^2)$ could be limiting in some cases, and as mentioned before, assigning to a cluster cannot be changed later, which could be a problem because a good local decision might not be a good global decision. Agglomerative clustering techniques have problems with noise, outliers, non-convex shapes and have a tendency to break large clusters.

2.4.2.5 CURE

CURE (CLustering using Representatives) [39] is a clustering algorithm that uses a variety of different techniques to remedy the drawbacks, of an agglomerative clustering method.

CURE represents a cluster by multiple points from the cluster. The first point to be chosen is the furthest point from the centre of the cluster. The remaining points are chosen in order be the farthest points from each other. Selecting points this way we might obtain pretty good distribution of points over the cluster. However if a selected point is an outlier (which are the farthest points), the description of the cluster becomes imprecise. The number of chosen points in each cluster is a parameter c, and it works well for $c \geq 10$.

Once representatives are chosen, all points are shrunk towards the centre of the cluster by a factor $\alpha, 0 < \alpha < 1$. This helps to moderate the effect of outliers, the absolute value of moving each point will be bigger for points lying farther out.

As a clustering method uses hierarchical agglomerative clustering (see Section 2.4.2.2 for more details), with distance function counting the minimum of any two representative points between two clusters. However, this algorithm requires specification of k, the number of clusters we would like to find.

There are two phases of eliminating outliers. The first one occurs when approximately 1/3 of the desired k clusters is reached. Clusters, which are growing slowly, are removed because they might potentially contain just outliers. The second phase of elimination is done when the desired number of clusters are discovered. At this point all small clusters are removed.

Complexity The time complexity would be in the worst case $O(N^2 log\, N)$, when the dimensionality of data points is small, it can be further reduced to $O(N^2)$. For storing data a heap is used and the required space is $O(N)$.

Optimization CURE uses two techniques to speed up the process, because the time complexity is high. Firstly, it takes a random sample of points and performs a clustering.

This is followed by a final pass, that assigns each of remaining points to the cluster with the closest representative.

The second phase eventually occurs when the sample required for clustering is still too large. Then the Divide & Conquer approach is applied. The data points are divided into N/p parts, in each clustering is performed and then the results are merged.

CURE is more robust to outliers than a hierarchical agglomerative clustering algorithm, and also it manages to identify clusters that have a non-spherical shape and wide variances in size.

2.5 Probabilistic Models for Clustering

Model-based clustering algorithms try to optimize the fit between the observed data and some mathematical model built using probabilistic approach. Typically such algorithm expect each cluster to follow certain distribution, such as Gaussian distribution or a mixture of underlying distributions. Then the clustering problem is transformed into a parameter estimation problem since the entire data could be modeled by a mixture of K components.

Suppose our dataset $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ is formed by a D dimensional random variable $\boldsymbol{x}$. The random variable $\boldsymbol{x}_n$ is assumed to be distributed according to mixture of K components (i.e. clusters). Formally the probability density function of $\boldsymbol{x}_n$ can be written as:

$$p(\boldsymbol{x}_n) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_n|\theta_k) \tag{2.20}$$

where $\pi_1, \pi_2, \ldots, \pi_k$ are the mixing probabilities, each θ_k is the set of parameters specifying the k^{th} component and $p(\boldsymbol{x}_n|\theta_k)$ is the component distribution. In order to have valid probabilities $\{\pi_k\}$ must satisfy:

$$0 \leq \pi_k \leq 1 (k = 1, \ldots, K) \tag{2.21}$$

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.22}$$

We need to infer a set of parameters from the observation, including the mixing probabilities $\{\pi_k\}$ and the parameters for component distributions $\{\theta_k\}$. Generally number of clusters k is considered as fixed parameter however some algorithms estimate k as part of integral part of model training [40]. The overall parameters for mixture model are: $\Theta = \{\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K\}$. Assuming that the data points are drawn independently from the distribution, then the probability of generating all the data points is:

$$p(\boldsymbol{X}, \Theta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_n|\theta_k) \tag{2.23}$$

Maximum likelihood estimation (ML) [21, 41] is an important statistical approach for parameter estimation:

$$\Theta_{ML} = \arg\max_{\Theta} \{\log p(\boldsymbol{X}|\Theta)\} \tag{2.24}$$

which considers the best estimate the one that maximizes the probability of generating all observations. The ML estimate cannot be found analytically [40]. The same applies for the Bayesian maximum a posteriori (MAP) criterion:

$$\Theta_{MAP} = \arg\max_{\Theta} \{\log p(\boldsymbol{X}|\Theta) + \log p(\Theta)\} \tag{2.25}$$

that requires a priori information $p(\Theta)$ about the parameters.

2.5.1 Gaussian Mixture Model

Gaussian Mixture model (GMM) [42] represent each component as normal distribution. For a single variable x the Gaussian distribution can be written as:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{2.26}$$

where μ is the mean and σ^2 is the variance.

For a D-dimensional vector $\boldsymbol{x}$ the multivariate Gaussian distribution takes form:

$$\mathcal{N}(x|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{2.27}$$

where $\boldsymbol{\mu}$ is a D-dimensional vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes determinant of $\boldsymbol{\Sigma}$.

In the Gaussian mixture model, each component is represented by the parameters of a multivariate Gaussian distribution $p(\boldsymbol{x}_n|\theta_k) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}_k)$. Based on Eq. 2.20, the Gaussian mixture distribution can be written as:

$$p(\boldsymbol{x}_n|\Theta) = p(\boldsymbol{x}_n|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.28}$$

Then the log-likelihood function is given by:

$$\ell(\Theta) = \log p(\boldsymbol{X}, \Theta) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\Theta) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.29}$$

We introduce a K-dimensional random variable $\boldsymbol{z}_n$ that has a 1-of-K representation, where one element z_{nk} equals to 1 and all other are set to 0. Using Bayes' theorem, we can obtain the conditional probability of $z_{nk} = 1$ given $\boldsymbol{x}_n$:

$$p(z_{nk}=1|\boldsymbol{x}_n) = \frac{p(z_{nk}=1)p(\boldsymbol{x}_n|z_{nk}=1)}{\sum_{j=1}^{K} p(z_{nj}=1)p(\boldsymbol{x}_n|z_{nj}=1)} = \frac{\pi_k p(\boldsymbol{x}_n|\theta_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_n|\theta_j)} \tag{2.30}$$

In order to find maximum likelihood solutions that are valid at local maximum, the derivatives of $\log p(\boldsymbol{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ respectively needs to be computed [30]:

$$\frac{\partial \ell}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{n=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = \sum_{n=0}^{N} \gamma(z_{nk}) \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) \tag{2.31}$$

By setting derivative to zero and multiplying by $\boldsymbol{\Sigma}_k$ we obtain:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.32}$$

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.33}$$

Similarly, we set the derivative of $\log p(\boldsymbol{X}, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to 0 and we get:

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\intercal}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.34}$$

Values of π_k are constrained to be positive and sum to one, this might be handled using a Lagrange multiplier. After simplifying we get [30]:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N} \tag{2.35}$$

so that the mixing probabilities for the k^{th} component are given by the average responsibility that the component takes for explaining the data points.

Maximizing the log-likelihood function for a Gaussian mixture model turns out to be a very complex problem. An elegant solution is called Expectation-Maximization algorithm [21] that is outlined in Algorithm 2.7.

The algorithm starts by initialization with guesses about means, covariances and mixing probabilities. Then alternates between the *expectation* step and *maximization* step. In E-step current parameters are used to calculate the posterior probabilities (responsibilities), while in M-step the log-likelihood is maximized using the updated responsibilities, also means, covariances and mixing probabilities are being re-estimated.

Compared to k-means, the EM algorithm requires much more iterations [43]. It is common to run k-means in order to find suitable initialization and speed-up EM algorithm convergence.

Algorithm 2.7 EM for Gaussian Mixture Models

1: **procedure** EM-GMM(*dataset*)
2: Initialize μ_k^0, $\mathbf{\Sigma}_k^0$, π_k^0
3: **while** convergence criterion not reached **do**
4: E-step: calculate $\gamma(z_{nk})$ using current parameters based on Eq.2.33
5: M-step: update μ_k using Eq. 2.32, then calculate $\mathbf{\Sigma}_k^0$ using Eq. 2.34 and finally re-estimate π_k^0 with Eq. 2.35, calculate log-likelihood using Eq. 2.28
6: **end while**
7: **end procedure**

2.6 Density-based methods

Methods developed on the notion of density. Typically clusters are considered as dense regions of objects in the feature space that are separated by regions with relatively low density. These methods are capable of discovering clusters of arbitrary shapes. Unfortunately, in general the amount of data required to estimate the local density at a given level of accuracy increases exponentially with the number of data dimensions [44]. This phenomena is well known as the curse-of-dimensionality.

2.6.1 DBSCAN

One of first approaches that measures similarity by the number of shared neighbors was proposed by [45].

The later DBSCAN[2] algorithm [46] takes a very similar approach; the algorithm is capable of discovering arbitrarily shaped clusters if the cluster density can be determined beforehand and each cluster has a uniform distribution. A cluster is defined as a maximum set of density-connected points, where every core point in a cluster must have at least a minimum number of points (*minPts*) within a given radius (ε).

All points within one cluster can be reached by traversing a path of density-connected points. The algorithm itself can be relatively fast, however in order to configure its parameters properly, prior knowledge of the dataset or landmarking by k-NN algorithm is required. DBSCAN is capable of detecting noise in data, an example is shown on Figure 2.3. The main disadvantage of DBSCAN is its sensitivity to parameter values: even a small modification of the ε parameter could cause all data points to be assigned to a single cluster. Moreover, the algorithm will fail on datasets with non-uniform distribution (as illustrated in Figure 2.6.1). The original paper incorrectly states $\mathrm{O}\left(n \log(n)\right)$ time complexity that could be achieved only in 2D space. For $d \geq 3$ the problem requires $\Omega(n^{4/3})$ time to solve as [47] showed. Later a hierarchical (and improved) version called HDBSCAN [48, 49] was proposed, that provides a complete density-based hierarchy from which most significant clusters can be extracted.

[2] A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise

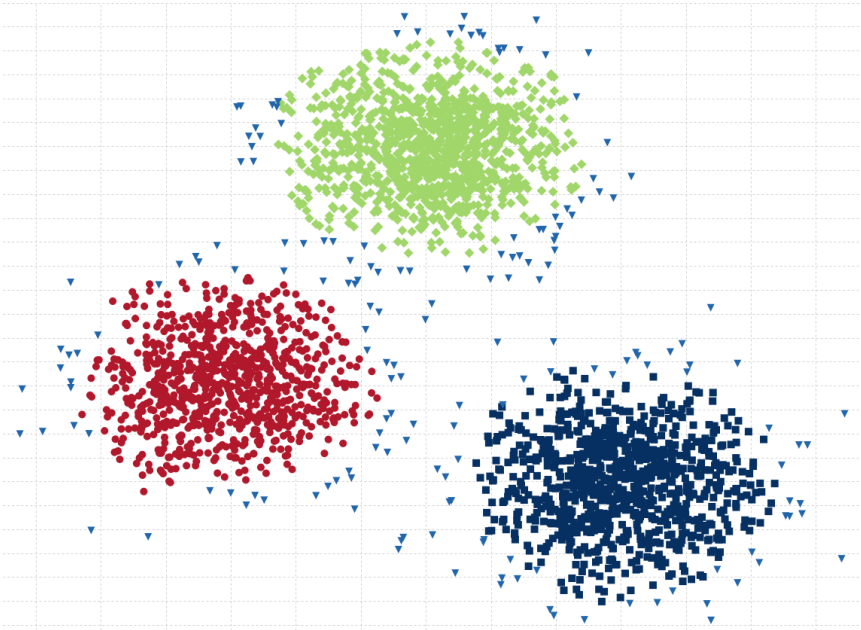


Figure 2.3: A successful DBSCAN clustering ($MinPts = 20$, $\varepsilon = 5$) on *xclara* dataset. Data points marked with triangles are considered as outliers. The algorithm is very sensitive to even small modification of parameters' values.

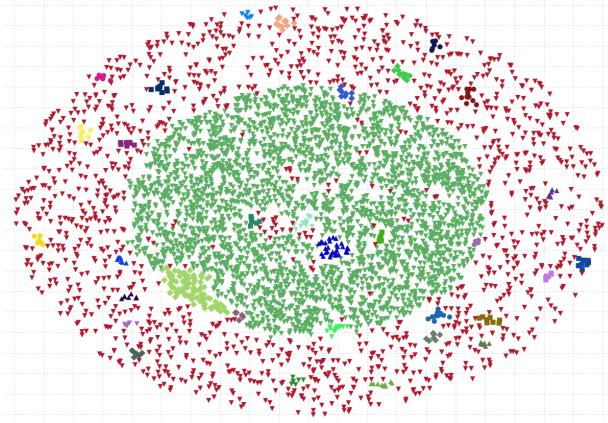


Figure 2.4: DBSCAN clustering of *dense-disk-5k* dataset with best configuration found ($minPts = 5, \varepsilon = 0.28$) includes many nonsense clusters, because DBSCAN cannot handle different densities of clusters.

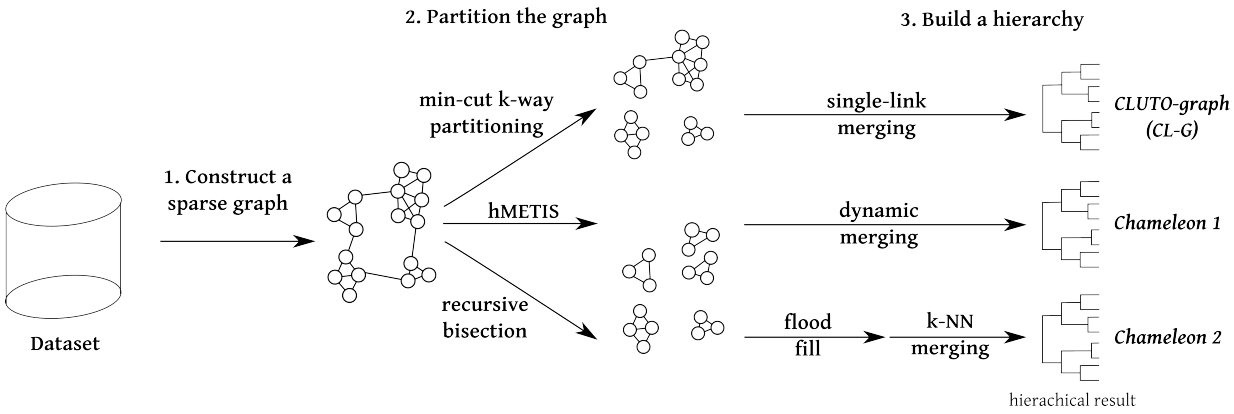


Figure 2.5: An overview of Chameleon family approaches. All these algorithms could be considered hybrid in terms of the traditional division between agglomerative and partitioning algorithms. These algorithms combine the partitioning phase with the agglomerative phase while producing a hierarchical result.

2.7 Graph based methods

Graph based methods are popular not only in domains where data is represent as a network, such as social network analysis or graph databases. Any dataset could be converted to a graph by connecting data points with their closest neighbors (basically running K-NN algorithm). In that case graph based method might be considered very similar to density based approaches as constructed graph is just another way of representing local density.

2.7.1 Chameleon

The Chameleon algorithm [28] operates on a graph representation of data. At first, we construct a graph based on k nearest neighbors. A data point, represented by a node in graph, is connected with its k closest neighbors by an edge that is given weight equal to inversion of their distance. If a graph is provided, we can skip this step and continue with the second one.

The second step *partitions* the previously created graph. The goal of partitioning is to produce equal-sized partitions and minimize the number of edges cut. In this manner, many small clusters are formed with a few highly connected nodes in each cluster. For partitioning, Karypis et al. use their own hyper-graph partitioning algorithm called hMETIS, which is a multilevel partitioning algorithm working with a coarsened version of the graph. Coarsening is non-deterministic, thus for each run we might obtain a slightly different result (for further details see [50] and [51]).

The final and most important step *merges* the partitioned clusters using Chameleon's dynamic modeling framework, starting with the most similar pair of clusters. This procedure is common for all agglomerative algorithms: the most similar pair of all possible

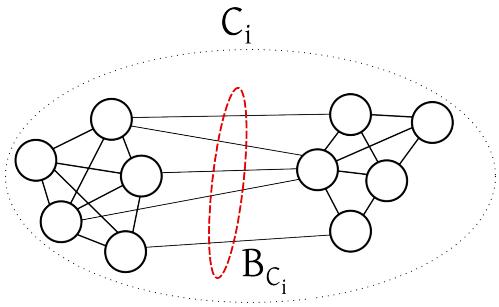


Figure 2.6: All nodes in this graph belong to a single cluster C_i, in order to compute bisection $\phi(C_i)$ we are looking for a set of edges B_{C_i} (marked with dashed line) whose removal would split the cluster into two roughly equal parts.

pairs of clusters is selected and merged to form a larger cluster. There are $n \cdot (n-1)/2$ such pairs, thus the time and memory complexity is $\mathrm{O}(n^2)$, however thanks to the previous partitioning we are merging clusters containing multiple items, so the number of merges is far smaller than in traditional hierarchical clustering[3]. There are many ways of computing similarities between clusters, e.g. based solely on the distance between two points (*single-link*, *complete-link*, the distance from a cluster's centroid, etc.) or using multiple points (e.g. *average-link*). Chameleon's approach belongs to the latter category, but in addition to computing distances between clusters it also accounts for intra-cluster distances. This idea is not entirely new; there are many examples between cluster validation metrics (such as [52] or [53]) which compute the ratio between cluster compactness and cluster separation. Nonetheless, in Chameleon both similarity metrics are computed during the merging phase, while in other approaches evaluation metrics are typically used for external result evaluation only.

During merging, Chameleon tries to merge clusters of similar densities. Density estimation relies on graph bisection, i.e. the division of a cluster (graph) into two roughly equal parts by cutting a minimal number of edges. The quality of the bisection algorithm used has major impact on merging order thus influences final clustering result.

Firstly we introduce a notation used for similarity metrics definition, each cluster is represented as a graph. $B_{C_i} = bisect(C_i)$ is a set of edges selected by a bisection algorithm. $w(e)$ is the weight (typically Euclidean distance inversion) of a given edge that was computed by the k-NN algorithm.

$$\phi(C_i) = \sum_{e \in B_{C_i}} w(e) \tag{2.36}$$

$$\bar{\phi}(C_i) = \frac{1}{|B_{C_i}|} \phi(C_i) \tag{2.37}$$

[3]The actual speed-up depends on the number of neighbors in the initial graph (parameter k) and the separation of items in the dataset. In case the dataset contains many isolated points, after partitioning each of those points will form an individual cluster, which has to be compared to all other clusters.

Where $\phi(C_i)$ represents the sum of the edges' weights in B_{C_i} and the average edge weight is noted as $\bar{\phi}(C_i)$ (an example is shown in Figure 2.6).

The between-cluster metrics are defined in a similar manner: the sum of the between-cluster edges' weights $s(C_i, C_j)$ and the average weight of all edges between clusters $\bar{s}(C_i, C_j)$, except that computing a bisection is not necessary in this case, because we already have two clusters.

The first measure employed by Chameleon is called *relative closeness*:

$$R_{CL}(C_i, C_j) = \frac{\bar{s}(C_i, C_j)}{\frac{|E_{C_i}|}{|E_{C_i}|+|E_{C_j}|}\bar{\phi}(C_i) + \frac{|E_{C_j}|}{|E_{C_i}|+|E_{C_j}|}\bar{\phi}(C_j)} \tag{2.38}$$

Where $|E_{C_i}|$, respective $|E_{C_j}|$ is the number of edges inside cluster C_i, respective C_j. The numerator is similar to the *average link* method, but only distances between nodes connected by an edge are included. The denominator uses average inter-cluster distances, which are normalized by the relative number of edges in each cluster (again an edge must be present between pairs in a cluster, in order to be included in this denominator). Using the same notation we define *relative interconnectivity*:

$$R_{IC}(C_i, C_j) = \frac{s(C_i, C_j)}{\frac{\phi(C_i)+\phi(C_j)}{2}} \tag{2.39}$$

High values of interconnectivity indicate that clusters C_i and C_j should be merged. The sum of the edges' weight between clusters is compared to the sum of the weights after cluster bisection. This normalization drives Chameleon to merge clusters with similar densities.

Finally, the *Ch1* similarity function between two clusters (C_i and C_j) is computed as [28]:

$$S_{\text{Ch1}}(C_i, C_j) = R_{CL}(C_i, C_j)^{\alpha} \cdot R_{IC}(C_i, C_j)^{\beta} \tag{2.40}$$

where α, β are user specified parameters that give higher importance either to relative closeness (compact clusters) or to relative interconnectivity (well-separated clusters). Default values are $\alpha = 2.0, \beta = 1.0$.

2.8 Multi-objective clustering

Multi-objective clustering usually optimizes two objective functions. Using more than a few objectives is not usual because the whole process of optimization becomes less effective.

The first multi-objective evolutionary clustering algorithm was introduced in 2004 by Handl and Knowles [55] and is called VIENNA (the Voronoi Initialized Evolutionary Nearest-Neighbour Algorithm).

Subsequently, in 2007 Handl and Knowles published a Pareto-based multi-objective evolutionary clustering algorithm called MOCK [56] (Multi-Objective Clustering with automatic K-determination). Each individual in MOCK is represented as a directed graph which is then translated into a clustering. The genotype is encoded as an array of integers whose length is same as the number of instances in the dataset. Each number is a pointer to another instance (an edge in the graph), since it is connected to the instance at a given index. This easily enables the application of mutation and crossover operations.

As a Multi-Objective Evolutionary Algorithm (MOEA), MOCK employs the Pareto Envelope-based Selection Algorithm version 2 (PESA-II) [57], which keeps two populations, an internal population of fixed size and a larger external population which is exploited to explore good solutions. Two complementary objectives, *deviation* and *connectivity*, are used as objectives in the evolutionary process.

A clear disadvantage of MOCK is its computation complexity, which is a typical characteristic of evolutionary algorithms. Nevertheless, the computation time spent on MOCK should result in high-quality solutions. Faceli et al. [58] reported that for some high-dimensional data it is not guaranteed that the algorithm will complete, unless the control front distribution has been adjusted for the given data set.

Faceli et al. [58] combined a multi-objective approach to clustering with ensemble methods and the resulting algorithm is called MOCLE (Muli-Objective Clustering Ensemble Algorithm). The objectives used in the MOCLE algorithm are the same as those used in MOCK [56]: deviation and connectivity. Unlike MOCK, in this case the evolutionary algorithm used is NSGA-II [59].

2.9 User's guidance

There are many possible ways of clustering a dataset, choosing the one which would match user's expectation is almost impossible. Several approaches has been introduced to tackle this problem. A semi-supervised methods might use a trained classification algorithm [60], constraint-based restrictions [61] or seeding and constraints [62].

Constraint-based For constraint clustering algorithms an initial background knowledge must be provided, which is afterwards used during clustering process. There are two-types of constraints:

- *must-link* – two instances have to be together in the same cluster
- *cannot-link* – two instances must be in different clusters

Generated partitioning will satisfy all given constraints, if it is possible. Wangstaff et al. introduced a constrained version of k-Means called COP-KMeans [61].

A problem with constrains might arise when specified constrains are contradictory. According to [63], algorithms preform better with the lower number of constrains, in many cases constrains might decrease accuracy.

2.10 Stability of clustering

A common problem with many clustering algorithms is sensitivity to small changes in input values which might results in a major reordering of the clustering result. To overcome problems connected to this issue several approaches has been proposed.

Consensus clustering is used for obtaining stable clusterings. The problem of unstable clusters arises especially with random initialization of an algorithm. But also a small change of parameter could cause significant changes in resulting clustering. Usually there are many objects that tends to stay together. These objects will produce stable clusters and the rest of items might end up in a noisy cluster or their assignment could be almost random.

Ensemble clustering is tightly connected to consensus clustering. An ensemble might be composed of several different algorithms or the same algorithm but its parameters will vary. To obtain a final solution we need a consensus function.

2.11 Scalable clustering algorithms

Scalability is an important issue for data-mining and in last years is getting even more attention. There are basically three aspects that could be considered. Firstly we have scalability in terms of records number N. For large datasets records has to be kept in secondary memory and therefore it is desirable to minimize disk accesses.

Secondly we can scale to number of dimensions d. High-dimensional spaces have strange properties which causes problems to methods that reliably perform in lower dimensions.

Lastly there is an aspect of scalability to number of processors, P. Ideally the amount of data could be split into P near equal parts that could be executed in parallel with nearly linear speedup.

Clustering algorithms are not inherently scalable, mainly because of computing distances between all points or at least all clusters can not simply use divide and conquer principle. The problem of large datasets has been approached in many ways, for example CURE [39] is using only a random sample of data set in order to maintain scalability.

In the literature probably most attention was paid to improving the popular k-means algorithm [20]. When the number of clusters k and number of dimensions d is fixed the complexity is

$$O\left(n^{dk+1}\log(n)\right)$$

where n is the number of objects to be clustered [64]. The "standard" version of algorithm which is usually referenced as k-means is the Lloyd's implementation [22]. It starts with random initialization of exactly k clusters, which is a parameter of the algorithm.

It has been shown that after first few iterations of the algorithm centroids become pretty much stable and their position in space does not change much. With usage of

triangular inequality Elkan managed to significantly speed up k-means algorithm [25]. Due to elimination of unnecessary computation of distances from each single point to cluster centers which are located far away (and therefore it's not likely that a point would we assigned to that cluster). This approach was later improved by Hamerly [65], at the expense of keeping more boundaries in memory many computation were eliminated and especially on higher dimensional datasets ($d > 100$) significantly outperforms other implementations.

2.12 Most challenging clustering problems

There are many challenging problems in clustering field that researchers have been trying to tackle during past decades. Part of these issues are described in Jain's "Data Clustering: 50 Years Beyond K-Means" [66].

- Clustering algorithm will always cluster data no matter whether there is a natural grouping or not.
- While clustering belongs to an unsupervised learning ("learning without a teacher") group where there is no teacher to tell us which assignment is correct. At least approximated guidance by an evaluation function for clustering results could improve clustering quality significantly.
- Two or three dimensional data can be easily inspected by a human. However in higher dimensions humans fails miserably. There is a need for a projection or dimensionality reduction that might deteriorate distances between objects.

Chapter 3

Chameleon 2

Although Chameleon [28] is considered to be one of the best hierarchical clustering algorithms [30], our experiments reveal several deficiencies in the final clustering. The main drawback of the algorithm is its inability to handle singleton clusters (clusters containing one or few items). Moreover, unlike DBSCAN and CURE, Chameleon does not handle noise at all. To overcome these issues we developed a Chameleon 2 algorithm [54]. A Chameleon 2 result is shown on Figure 3.1.

3.1 Refined Similarity

Original Chameleon has an issue with many disconnected clusters in low density regions after hMETIS partitioning. The solution is contaminated with many singleton clusters that should be assigned to neighboring clusters instead of forming individual clusters. This is not solely an issue of partitioning, but also a sign that the similarity metric does not successfully capture the notion of a point's neighborhood, because close points are not merged together. Most problematic is the R_{IC} definition. The sum of bisected edges penalizes clusters that can be easily divided into two parts.

Instead of relying on metrics computed from bisection we base similarity on the ratio between the average weight of all edges across two clusters and the sum of weights inside those clusters, denoted as:

$$s(C_i) = \sum_{e \in C_i} w(e) \tag{3.1}$$

$$\bar{s}(C_i) = \frac{1}{|E_{C_i}|} s(C_i) \tag{3.2}$$

and $\bar{s}(C_i, C_j)$ is computed in the same manner using the average of the inter-cluster edge weights between cluster C_i and C_j. Thus, closeness is similar to the original one in Equation 2.38:

$$R_{CL2}(C_i, C_j) = \begin{cases} m_{\text{fact}} \cdot \dfrac{\bar{s}(C_i, C_j)}{s(C_i) + s(C_j)} & \text{for } |E_{C_i}| \vee |E_{C_j}| = 0 \\ \left(|E_{C_i}| + |E_{C_j}|\right) \cdot \dfrac{\bar{s}(C_i, C_j)}{s(C_i) + s(C_j)} & \text{for } |E_{C_i}| \wedge |E_{C_j}| > 0 \end{cases} \tag{3.3}$$

Modified interconnectivity considers only the ratio between the number of edges, instead of relying on edge weights (Equation 2.39):

$$R_{\text{IC2}}(C_i, C_j) = \begin{cases} 1 & \text{for } |E_{C_i}| \vee |E_{C_j}| = 0 \\ \dfrac{|E_{C_{i,j}}|}{\min\left\{|E_{C_i}|, |E_{C_j}|\right\}} \cdot \rho(C_i, C_j)^{\beta} & \text{for } |E_{C_i}| \wedge |E_{C_j}| > 0 \end{cases} \tag{3.4}$$

Where $|E_{C_{i,j}}|$ represents the number of edges between clusters C_i and C_j. This part of the equation encourages merging of clusters connected by a large number of edges relative to the number of edges inside the clusters.

The last addition is the ρ factor, which discourages the algorithm from merging clusters with different densities: both clusters should have similar average between-point distances. The β parameter serves to modify the weight of the ρ factor.

$$\rho(C_i, C_j) = \frac{\min\{\bar{s}(C_i), \bar{s}(C_j)\}}{\max\{\bar{s}(C_i), \bar{s}(C_j)\}} \tag{3.5}$$

For both interconnectivity and closeness we have to handle a special case: a cluster containing an individual node. In that case, the cluster contains no edges, which leads to zero similarity. Such clusters are often merged incorrectly, worsening the overall result. Even when the partitioning algorithm is configured to make strictly balanced partitions with the same number of items in each cluster, this problem persists. Therefore, this issue has to be handled during the merging phase.

When computing the similarity of a cluster pair where one of the clusters contains no edges, we compute only the pair's R_{CL2}. Then we multiply the R_{CL2} by a constant m_{fact} to obtain the final cluster similarity. This process increases the similarity of all pairs containing single-item clusters and causes the clusters to merge with their neighbors in the early stages of the merging process, avoiding problems later on.

The resulting similarity is a product of modified closeness and interconnectivity – as in the case of the original Chameleon. Then Chameleon 2 base similarity is defined as:

$$S_{\text{Ch2}}(C_i, C_j) = R_{\text{CL2}}(C_i, C_j)^{\alpha} \cdot R_{\text{IC2}}(C_i, C_j) \tag{3.6}$$

Another problem is the absence of an algorithm for the extraction of high-quality partitioning from hierarchical structures created by the merging process. This problem is common to all agglomerative algorithms. In order to overcome this issue, we developed a heuristic for flat partitioning extraction that is called *First Jump*. A detailed description can be in [54].

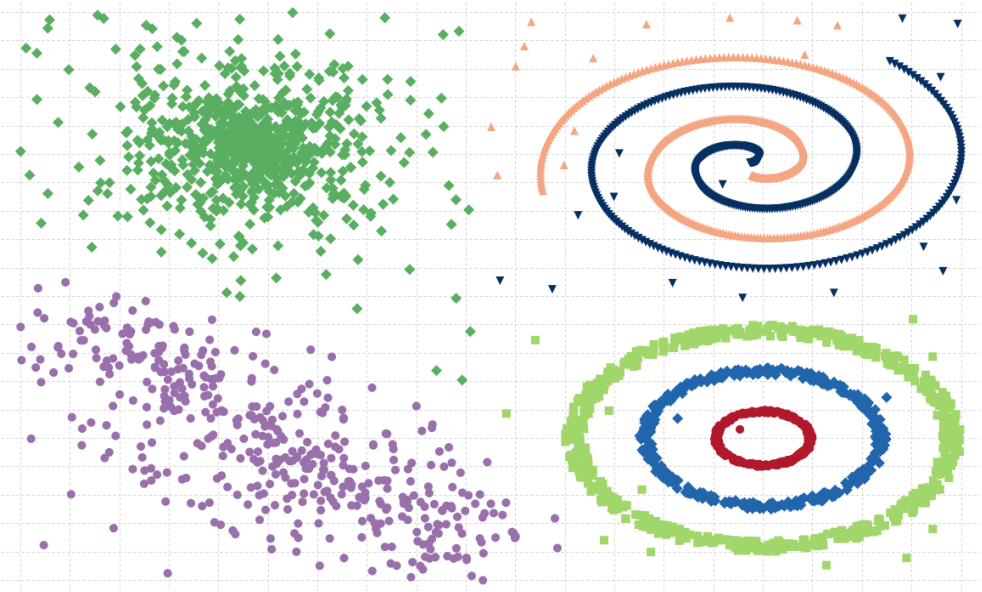


Figure 3.1: A Chameleon 2 result on a toy dataset, inspired by Jain et al., who suggest that it cannot be solved by a clustering algorithm. Our proposed algorithm gives a solution very close to human judgment.

3.2 Summary

Chameleon 2 corrects issues present in previous version and pushes further boundaries of single-run clustering algorithms aiming for high-quality partitions. The algorithm performs well on complex datasets without necessity of fine-tuning its parameters. The main disadvantage is its complexity that comes from k-NN graph construction and consequent hierarchical merging. When Chameleon 2 is employed in more sophisticated pipeline the k-NN graph could be shared between multiple algorithms. Compared to classical hierarchical clustering Chameleon perform less merges as the smallest leaf in dendrogram already represents multiple points.

Chapter 4

Clustering evaluation

"Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

— Jain and Dubes *[1], 1988*

More striking than the statement above is the fact that it still holds true after 28 years [67]. Despite all the progress that has been made, quality assessment of models in unsupervised learning and clustering verification in particular have been a long-standing problem in the machine learning research [68].

Assuming we have a clustering $\mathbb{C}$ and yet another clustering $\mathbb{C}'$. How are we supposed to tell which one is better or whether they are the same? Clustering are typically stored as a list with cluster numbers, e.g. $\mathbb{C} = (1, 1, 2, 2, 3, 3)$ and $\mathbb{C}' = (3, 3, 1, 1, 2, 2)$ are supposed to be considered the same. This property is called relabeling invariance.

Cluster validation measures are typically divided into there categories [69]:

- **External Criteria** – Measure similarity of clusterings against a priori known class labels. Since real-world datasets do not have labels these criteria could be used for controlled experiments only.
- **Internal Criteria** – Measure partitioning quality based e.g. on data distribution, distances to cluster centers, distances between clusters, etc.
- **Relative Criteria** – Compare two clusterings in order to determine their similarity in relative terms. Most external criteria meet this requirement, but the term *relative validity criteria* often refers to internal criteria that are also relative [70].

In this text we use following conventions:

Feature vector or an **instance** is a set of attributes which are used by clustering algorithm for grouping

$\mathbb{C} \setminus \mathbb{C}'$	C'_1	C'_2	...	C'_l	Sums
C_1	M_{11}	M_{12}	...	M_{1l}	u_1
C_2	M_{21}	M_{22}	...	M_{2l}	u_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_k	M_{k1}	M_{k2}	...	n_{kl}	u_k
Sums	v_1	v_2	...	v_l	$\sum_{ij} M_{ij} = N$

Table 4.1: Contingency Table (also called Confusion Matrix)

Attribute (or **feature**) is one value from the feature vector which is represented by a number[1]

Distance measure is a metric for computing distance between two feature vectors

Let X be a finite set of instances with cardinality $|X| = N$. A clustering $\mathbb{C}$ is a set $\{C_1, C_2, \ldots, C_k\}$ of non-empty subsets of X, so that the union is equal to X:

$$X = \bigcup_{i=1}^{k} C_i$$

Each vector of X has D attributes (features). This could be represented by a data matrix $Z_{N \times D}$.

Let $\mathbb{C}' = \{C'_1, \ldots, C'_l\}$ denote a second clustering of X with number of clusters equal to l. The *contingency table* (or *confusion matrix*) M of pairs $\mathbb{C}, \mathbb{C}'$ is a $k \times l$ matrix whose ij-th entry equals the number of elements in the intersection of clusters C_i and C'_j:

$$M_{ij} = \left| C_i \cap C'_j \right|, \; 1 \leq i \leq k, \; 1 \leq j \leq l$$

as shown in Table 4.1.

4.1 External Validation Criteria

From definition in unsupervised learning such as clustering we should not have labels. Nonetheless in some cases like verification of an algorithm these labels are necessary. Then we can use traditional metrics for effectiveness, *recall* and *precision*. Recall signifies the proportion of correct assignments in result clustering. Let A be a set of "correct" clustering and B is a set of clustering produced by an algorithm. What we get is criteria for supervised learning [34]:

[1] generally also strings or boolean values are acceptable however it is out of scope of this work

	actual class	
predicted class	True Positive S_{11} *correct result*	False Positive S_{01} *incorrect result*
	False Negative S_{10} *missing result*	True Negative S_{00} *correct absence*

Table 4.2: Classification categories used in supervised learning and our corresponding sets

$$recall = P(B|A) = \frac{P(A \cup B)}{P(A)} = \frac{n_{11}}{n_{11} + n_{10}} = \frac{TP}{TP + FN}$$

$$precision = P(A|B) = \frac{P(A \cup B)}{P(B)} = \frac{n_{11}}{n_{11} + n_{01}} = \frac{TP}{TP + FP}$$

Precision and recall are frequently used in other fields like information retrieval, supervised learning or recommender systems.

4.1.1 Counting pairs

A very intuitive approach of comparing clusterings is counting pairs of instances in each clustering. We can divide our set of instances into 4 subsets which are specified in Table 4.3

S_{11}	a	TP	pairs in the same cluster under $\mathbb{C}$ and $\mathbb{C}'$
S_{10}	b	FP	pairs in the same cluster under $\mathbb{C}$ and in different under $\mathbb{C}'$
S_{01}	c	FN	pairs in the same cluster under $\mathbb{C}'$ and in different cluster under $\mathbb{C}$
S_{00}	d	TN	pairs in different cluster under both $\mathbb{C}$ and $\mathbb{C}'$

Table 4.3: All possibilities of pairs placement when comparing two clusterings. Notation with TP (true positive) is sometimes also used.

With cardinality $|X| = N$, the number of different pairs we can choose from a set is

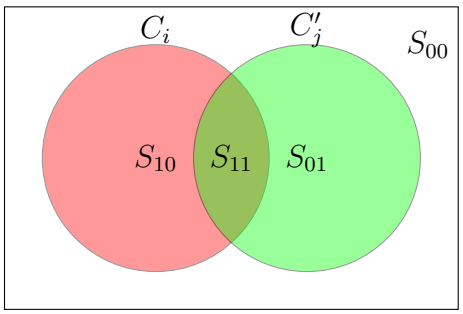


Figure 4.1: When comparing cluster C_i in clustering $\mathbb{C}$ with cluster C'_j in clustering $\mathbb{C}'$ four cases can occur: an item is assigned to both clusters (S_{11}), an item is in cluster C_i but not in C'_j (S_{10}), an item is not in C_i but it is in C'_j (S_{01}) and lastly item is not in any of clusters (S_{00}).

given by the combination number:

$$\begin{aligned}\binom{n}{2} &= |S_{11}| + |S_{10}| + |S_{01}| + |S_{00}| \\ &= a + b + c + d\end{aligned}$$

4.1.1.1 Chi Squared (χ^2) Coefficient

This is probably the oldest measure coefficient widely used by statisticians. It is defined as:

$$\chi(\mathbb{C}, \mathbb{C}') = \sum_{i=1}^{k}\sum_{j=1}^{l} = \frac{(M_{ij} - E_{ij})^2}{E_{ij}}$$
$$\text{where} \quad E_{ij} = \frac{|C_i||C'_j|}{n}$$

Originally it was suggested by Pearson in 1900 for testing independence in a bivariate distribution, not for evaluating association.

The problem of using such a measure for comparing clusterings is the fact that we have to assume independence of the two clusterings. In general this is not true, especially when we compare two clusterings produced by the same algorithm but with different parameters [71].

4.1.1.2 Rand Index

General Rand Index This index for originally proposed for measuring precision of supervised classification algorithms. Where the fraction between correctly classified (respectively misclassified) elements to all elements is calculated. For comparing clusterings

we count pairs of elements instead of single elements. The Rand Index is defined as [72]:

$$\text{Rand}(\mathbb{C}, \mathbb{C}') = \frac{2(S_{11} + S_{00})}{n(n-1)} = \frac{S_{11} + S_{00}}{S_{11} + S_{10} + S_{01} + S_{00}} = \frac{S_{11} + S_{00}}{\binom{n}{2}}$$

When no pair is classified the same way under both clustering Rand has value 0. For identical clusterings it would be 1. However the Rand index converges to 1 as the number of clusters is increased which is undesirable for a similarity measure [73]. Therefore the adjusted version was proposed by Hubert and Arabie.

Adjusted Rand Index The Adjusted Rand Index [74] assumes a generalized hypergeometric distribution as null hypothesis: the two clusterings are drawn randomly with a fixed number of clusters and a fixed number of elements in each cluster (the number of clusters in each clustering need not be the same). Then the Adjusted Rand Index is the (normalized) difference of the Rand Index and its expected value under the null hypothesis. It is defined as follows:

$$\text{AdjRand}(\mathbb{C}, \mathbb{C}') = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l}\binom{m_{ij}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3}$$

$$\text{where} \quad t_1 = \sum_{i=1}^{k}\binom{|C_i|}{2},$$

$$t_2 = \sum_{j=1}^{l}\binom{|C'_j|}{2},$$

$$t_3 = \frac{2t_1 t_2}{n(n-1)}$$

The significance of this measure has to be put into question because of the strong assumption it makes on the distribution [71]. An undesired property is that ARI can take negative values. For perfect match $\text{AdjRand}(\mathbb{C}, \mathbb{C}') = 1$.

Also following definition is possible:

$$\text{AdjRand}(\mathbb{C}, \mathbb{C}') = \frac{S_{10} + S_{01}}{S_{11} + S_{10} + S_{01} + S_{00}}$$

4.1.1.3 Jaccard Index

The Jaccard Index [75] is very similar to the Rand Index, however it disregards the number of elements that are in different clusters. It is defined as follows:

$$J(\mathbb{C}, \mathbb{C}') = \frac{S_{11}}{S_{11} + S_{10} + S_{01}} \tag{4.1}$$

or definition based on sets:

$$J(\mathbb{C}, \mathbb{C}') = \frac{|\mathbb{C} \cap \mathbb{C}'|}{|\mathbb{C} \cup \mathbb{C}'|}$$

4.1.1.4 Folkes and Mallows

Folkes and Mallows [73] proposed yet another evaluation metric based on counting pairs. The index is directly proportional to the number of true positives (S_{11}), higher index value means greater similarity between two clusterings.

$$\begin{aligned} m_1 &= S_{11} + S_{10}, \\ m_2 &= S_{01} + S_{00}, \\ p_{fm} &= \frac{a}{\sqrt{m_1 \cdot m_2}} = \frac{S_{11}}{\sqrt{(S_{11} + S_{10}) \cdot (S_{01} + S_{00})}} \end{aligned}$$

With growing number of cluster the index converges to 0 (unlike e.g. Rand Index from Section 4.1.1.2). The index is the geometric mean of the precision and recall, thus it is also known as the G-Measure, while the F-measure is their harmonic mean [74].

4.1.1.5 F-measure

F-measure (also known as F-score or F_1 score, the Sørensen–Dice coefficient or Dice similarity coefficient) balances precision P and recall R [76]:

$$F_1(\mathbb{C}, \mathbb{C}') = \frac{2PR}{P + R} = \frac{2S_{11}}{2S_{11} + S_{10} + S_{01}}$$

that was originally introduced by Rijsbergen [77] as effectiveness measure. The general formula for positive real β is:

$$F_\beta(\mathbb{C}, \mathbb{C}') = \frac{(1 + \beta^2)P \cdot R}{\beta^2 \cdot P + R}$$

where the other commonly used values are $\beta = 2$, which weights recall higher than precision and $\beta = 0.5$ that has the opposite effect.

4.1.1.6 Partition difference

The Partition Difference simply counts pairs elements that belong to different clusters under both clusterings:

$$\mathrm{PD}(\mathbb{C}, \mathbb{C}') = S_{00}$$

This measurement tries to express the difference of clustering however it does not capture number of elements placed in same cluster moreover is sensitive to clustering sizes and it's hard to interpret. In our opinion this measure should not be used at all.

4.1.1.7 Cluster difference

Cluster difference [6] is related to the Rand index, it counts pairs which are in the same cluster in clustering $\mathbb{C}$, but in different clusters under $\mathbb{C}'$ (or the other way round).

$$\text{ClusterDiff}(\mathbb{C}, \mathbb{C}') = \frac{2(S_{10} + S_{01})}{N \cdot (N-1)}$$

4.1.1.8 Mirkin metric

Mirkin metric [78] is another form of Rand Index. It could be expressed as:

$$M(\mathbb{C}, \mathbb{C}') = 2(S_{01} + S_{10}) = N \cdot (N-1)\,(1 - \text{Rand}(\mathbb{C}, \mathbb{C}'))$$

The metric corresponds to the Hamming distance for binary vectors if the set of all pairs of elements is enumerated and a clustering is represented by a binary vector defined on this enumeration [78].

4.1.2 Information Theory Based Criteria

Many clustering criteria comes from Information Theory background where the amount of shared information between two clusterings is computed. In order to compute such measure for external labels, we create a virtual (reference) clustering.

4.1.2.1 Normalized Mutual Information

Normalized Mutual Information (NMI) comes from Information Theory background [79] where it accounts for shared information. In clustering context NMI computes agreement between cluster assignments and ground truth labels. Given two clusterings $\mathbb{C}$ and $\mathbb{C}^{'}$, their entropies, joint entropy, conditional entropies and mutual information (MI) are defined via the marginal and joint distributions of data items in $\mathbb{C}$ and $\mathbb{C}^{'}$ respectively as:

$$H(\mathbb{C}) = -\sum_{i=1}^{K} \frac{u_i}{n} \log \frac{u_i}{N}, \tag{4.2}$$

$$H(\mathbb{C}, \mathbb{C}') = -\sum_{i=1}^{K} \sum_{j=1}^{K'} \frac{M_{ij}}{N} \log \frac{M_{ij}}{N}, \tag{4.3}$$

$$H(\mathbb{C}|\mathbb{C}') = -\sum_{i=1}^{K} \sum_{j=1}^{K'} \frac{M_{ij}}{N} \log \frac{M_{ij}/N}{v_j/N}, \tag{4.4}$$

$$I(\mathbb{C}, \mathbb{C}') = \sum_{i=1}^{K} \sum_{j=1}^{K'} \frac{M_{ij}}{N} \log \frac{M_{ij}/N}{u_i v_j/N^2}, \tag{4.5}$$

Where M is the contingency table and M_{ij} is the number of data points that are assigned to cluster C_i in clustering $\mathbb{C}$ and to C_j in clustering $\mathbb{C}'$.

Then NMI_{sqrt} is according to [80] defined as:

$$\mathrm{NMI}_{\mathrm{sqrt}}(\mathbb{C}, \mathbb{C}') = \frac{I(\mathbb{C}, \mathbb{C}')}{\sqrt{H(\mathbb{C})\, H(\mathbb{C}')}} \tag{4.6}$$

while [79] proposed several variants with diferent normalization. Most frequently $\mathrm{NMI}_{\mathrm{max}}$ and $\mathrm{NMI}_{\mathrm{avg}}$ are used:

$$\mathrm{NMI}_{\mathrm{max}}(\mathbb{C}, \mathbb{C}') = \frac{I(\mathbb{C}, \mathbb{C}')}{\max\{H(\mathbb{C}), H(\mathbb{C}')\}} \tag{4.7}$$

$$\mathrm{NMI}_{\mathrm{avg}}(\mathbb{C}, \mathbb{C}') = \frac{2I(\mathbb{C}, \mathbb{C}')}{H(\mathbb{C}) + H(\mathbb{C}')} \tag{4.8}$$

4.1.2.2 Variation of Information (VI)

Meilă [81] proposed another criterion VI that is defined as:

$$\mathrm{VI}(\mathbb{C}, \mathbb{C}') = H(\mathbb{C}|\mathbb{C}') + H(\mathbb{C}'|\mathbb{C}) \tag{4.9}$$

$$= 2H(\mathbb{C}|\mathbb{C}') - H(\mathbb{C}) \; - H(\mathbb{C}') \tag{4.10}$$

and is designed to measure lost and gained information in changing from clustering $\mathbb{C}$ to clustering $\mathbb{C}'$.

4.1.2.3 V-Measure

V-Measure [82] is another cluster evaluation measure based on entropy. The value is computed as weighted harmonic mean of two components named *completeness* and *homogenity*:

$$h(\mathbb{C}, \mathbb{C}') = \begin{cases} 1 & \text{if } H(\mathbb{C}, \mathbb{C}') = 0 \\ 1 - \frac{H(\mathbb{C}|\mathbb{C}')}{H(\mathbb{C})} & \text{otherwise} \end{cases} \tag{4.11}$$

$$c(\mathbb{C}, \mathbb{C}') = \begin{cases} 1 & \text{if } H(\mathbb{C}', \mathbb{C}) = 0 \\ 1 - \frac{H(\mathbb{C}'|\mathbb{C})}{H(\mathbb{C}')} & \text{otherwise} \end{cases} \tag{4.12}$$

$$V_\beta(\mathbb{C}, \mathbb{C}') = \frac{(1+\beta) \cdot h(\mathbb{C}, \mathbb{C}') \cdot c(\mathbb{C}, \mathbb{C}')}{(\beta \cdot h(\mathbb{C}, \mathbb{C}')) + c(\mathbb{C}, \mathbb{C}')} \tag{4.13}$$

Both VI and V-Measure satisfy desirable mathematical properties described by Dom [83]. With $\beta = 1$, V_β is highly correlated with NMI_{avg}.

4.2 Internal clustering validation

Firstly we introduce commonly used notation with several concepts that are commonly used in many clustering validation critera.

N_t ... total number of pairs of data points in the dataset X

$$N_t = \frac{N(N-1)}{2} \tag{4.14}$$

N_w ... total number of pairs of data points belonging to the same cluster

$$N_w = \sum_{k=1}^{K} \frac{n_k(n_k - 1)}{2} \tag{4.15}$$

N_b ... total number of pairs of observations belonging to different clusters

$$N_b = N_t - N_w \tag{4.16}$$

S_w ... sum of the within-cluster distances:

$$S_w = \sum_{k=1}^{K} \sum_{\substack{x_i, x_j \in C_k \\ i<j}} d(x_i, x_j) \tag{4.17}$$

S_b ... sum of the between-cluster distances:

$$S_b = \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} \sum_{\substack{x_i \in C_k \\ x_j \in C_l}} d(x_i, x_j) \tag{4.18}$$

$\mathbf{B}_q$ ... between-group dispersion matrix:

$$\mathbf{B}_q = B_{GSS} = \mathrm{Tr}(\mathbf{B}) = \sum_{k=1}^{K} n_k \left(\bar{\boldsymbol{x}} - \boldsymbol{c}_k\right) \left(\bar{\boldsymbol{x}} - \boldsymbol{c}_k\right)^\intercal \tag{4.19}$$

where $\bar{\boldsymbol{x}}$ is the overall dataset mean.
$\mathbf{T}$ ... total dispersion matrix is equal to N times the variance-covariance matrix.

$$\boldsymbol{T} = \boldsymbol{X}^\intercal \boldsymbol{X} \tag{4.20}$$

The general term of $\boldsymbol{T}$ can be written:

$$t_{ij} = N \times \mathrm{Cov}(V_i, V_j) \tag{4.21}$$

where V_j is the column vector ($1 \leq j \leq D$) of data matrix $\boldsymbol{X}$.
The total scattering $\boldsymbol{T}_{SS}$ (total sum of squares) is the trace of the matrix $\boldsymbol{T}$:

$$\boldsymbol{T}_{SS} = \mathrm{Tr}(\boldsymbol{T}) = N \sum_{j=1}^{D} \mathrm{Var}(V_j) \tag{4.22}$$

$\mathbf{W}$ ... within-group scatter matrix is a square symmetric matrix with dimensions $D \times D$ defined for each cluster:

$$\mathbf{W} = \sum_{k=1}^{K} \mathbf{W}^{\{k\}} \tag{4.23}$$

where the general term could be written as:

$$w_{ij}^{\{k\}} = \left(V_i^{\{k\}} - \mu_i^{\{k\}}\right) \left(V_j^{\{k\}} - \mu_j^{\{k\}}\right)^\intercal \tag{4.24}$$

or in terms of variance and covariance:

$$\begin{cases} w_{ij}^{\{k\}} & = n_k \times \mathrm{Cov}\left(V_i^{\{k\}}, V_j^{\{k\}}\right) \\ w_{ii}^{\{k\}} & = n_k \times \mathrm{Var}\left(V_i^{\{k\}}\right) \end{cases} \tag{4.25}$$

W_q ... within-group dispersion:

$$W_q = W_{GSS} = \mathrm{Tr}(\mathbf{W}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \boldsymbol{c}_k)(\boldsymbol{x}_i - \boldsymbol{c}_k)^\intercal \tag{4.26}$$

$$= \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \tag{4.27}$$

4.2.1 Akaike Information Criterion (AIC)

Akaike information criterion [84, 85] is typically used in supervised learning when trying to estimate model error. Basically it tries to estimate the optimism of the model with m parameters and then add it to the error [86].

$$\mathrm{AIC}(\mathbb{C}) = -2 \cdot \ln \mathcal{L}(\theta) + 2 \cdot m \tag{4.28}$$

The first term is a maximum likelihood of a model $\mathcal{L}(\theta)$ that stands for the penalty of badness of fit when the maximum likelihood estimators of the parameters of the model are used. The second term in the definition of AIC, on the other hand, stands for the penalty of increased unreliability or compensation for the bias in the first term as a consequence of increasing number of parameters [87].

There are multiple variants of AIC for cluster analysis. A simplified model might consider share covariance matrix $\boldsymbol{\Sigma}$ for all groups (clusters).

The likelihood for a clustering is given by:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{X}) = \prod_{g=1}^{K} \mathcal{L}_g(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g | \boldsymbol{X}_g) \tag{4.29}$$

The model assumes that data vectors are independent and identically distributed according to a multivariate normal distribution $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Where $\boldsymbol{\mu}$ is a d dimensional mean vector and $\boldsymbol{\Sigma}$ is a $d \times d$ covariance matrix. In terms of parameters the the multivariate analysis of variance (MANOVA) assumes $\theta = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \ldots, \boldsymbol{\Sigma}_K)$ with $m = Kd + d(d+1)/2$ parameters where K is the number of clusters and d is the data vector dimensionality.

The log likelihood function is given by [87]:

$$\ell(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g | \boldsymbol{X}_g) \equiv \log \mathcal{L}(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g | \boldsymbol{X}_g) \tag{4.30}$$

$$= \frac{Nd}{2} \log 2\pi - \frac{1}{2} \sum_{g=1}^{K} \ln |\boldsymbol{\Sigma}_g| - \frac{1}{2} \sum_{g=1}^{K} n_g \,\mathrm{Tr}\left(\boldsymbol{\Sigma}_g^{-1}\right) \boldsymbol{A}_g \tag{4.31}$$

$$- \frac{1}{2} \sum_{g=1}^{K} n_g \left(\mathbf{x}_g - \boldsymbol{\mu}_g\right)^\intercal \left(\mathbf{x}_g - \boldsymbol{\mu}_g\right) \tag{4.32}$$

The maximum likelihood estimation (MLE) of $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g$ are:

$$\widehat{\boldsymbol{\mu}}_g = \boldsymbol{x}_g,\ g = 1, 2, \dots K \tag{4.33}$$

$$\widehat{\boldsymbol{\Sigma}}_g = \frac{\boldsymbol{A}_g}{n_g} \tag{4.34}$$

Substituting these back and simplifying, the maximized likelihood becomes:

$$\ell(\widehat{\boldsymbol{\mu}}_g, \widehat{\boldsymbol{\Sigma}}_g | \widehat{\boldsymbol{X}}_g) \equiv \log \mathcal{L}(\widehat{\boldsymbol{\mu}}_g, \widehat{\boldsymbol{\Sigma}}_g | \boldsymbol{X}_g) \tag{4.35}$$

$$= -\frac{Nd}{2} \log 2\pi - \frac{1}{2} \sum_{g=1}^{K} n_g \log \left| \frac{\boldsymbol{A}_g}{n_g} \right| - \frac{Nd}{2} \tag{4.36}$$

Then AIC is becomes:

$$\mathrm{AIC}(\mathbb{C}) = Nd \ln(2\pi) + \sum_{g=1}^{K} n_g \ln \left| \frac{\boldsymbol{A}_g}{n_g} \right| + Nd + 2 \cdot \left(Kd + \frac{Kd(d+1)}{2} \right) \tag{4.37}$$

where n_g is the size of g-th cluster
$|\boldsymbol{A}_g|$ the determinant of matrix $\boldsymbol{A}_g = \sum_{x_i \in C_g} (\boldsymbol{x}_i - \boldsymbol{c}_g)(\boldsymbol{x}_i - \boldsymbol{c}_g)^\intercal$
$\boldsymbol{c}_g$ is a mean vector for cluster C_g
d is data dimensionality
K is number of clusters

A sum of $\boldsymbol{A}_g$ would give us within-group scatter matrix (defined in 4.23):

$$\mathbf{W} = \sum_{g=1}^{K} \boldsymbol{A}_g \tag{4.38}$$

AIC definition 4.37 becomes problematic in edge case clustering (e.g. clusters with single item) when determinant $|\boldsymbol{A}_g|$ is ≤ 0. The logarithm is not defined for such value which leads to an undefined value for the whole clustering. Therefore we introduce a correction:

$$\mathrm{I}(C_g) = \begin{cases} |n_g^{-1} \boldsymbol{A}_g| & \text{if } |\boldsymbol{A}_g| > 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.39}$$

Finally we get:

$$\mathrm{AIC}(\mathbb{C}) = Nd \ln(2\pi) + \sum_{g=1}^{K} n_g \ln\left(\mathrm{I}(C_g)\right) + Nd + 2 \cdot \left(Kd + \frac{Kd(d+1)}{2} \right) \tag{4.40}$$

AIC with varying $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ assumes normal distribution for all clusters which could be strong assumption for all data. Nonetheless it is a statistically solid criteria with penalty for small clusters.

On similar principle works also other statistics like C_p or BIC.

4.2.2 Bayesian Information Criterion (BIC)

Bayesian information criterion is suitable for application where the fitting is carried out by maximization of a log-likelihood. The BIC statistic (times 1/2) is also known as the Schwarz criterion [88].

$$\mathrm{BIC}(\mathbb{C}) = -2 \cdot \ln \mathcal{L}(\theta) + m \cdot \ln(N) \tag{4.41}$$

where $\ln \mathcal{L}(\theta)$ is the maximum likelihood of a model
with m parameters based on a sample of size n.

Using same approach as for AIC 4.40 we get:

$$\mathrm{BIC}(\mathbb{C}) = Nd \ln(2\pi) + \sum_{g=1}^{K} n_g \ln\left(\mathrm{I}(C_g)\right) + Nd + \cdot \left(Kd + \frac{Kd(d+1)}{2}\right) \ln(N) \tag{4.42}$$

BIC will select the correct model approaches one as the sample size $n \to \infty$. This is not the case for AIC, which tends to choose models which are too complex as $n \to \infty$. On the other hand, for finite samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity [86].

For clustering evaluation BIC gives higher penalty on high number of clusters, which makes BIC better choice for datasets where we prefer clusterings with few clusters.

4.2.3 TraceW index

Originally proposed by Edwards et al. [89], used to be one of the most popular indexes in clustering literature [90, 91, 92, 93] defined simply as:

$$\mathrm{TrW}(\mathbb{C}) = \mathrm{Tr}(\mathbf{W}) = W_{GSS} \tag{4.43}$$

The criterion increases monotonically with fewer clusters.

4.2.4 Ball-Hall index

The mean dispersion of a cluster is the mean of the squared distances of the points of the cluster with respect to their centroid. The Ball-Hall index is the mean through all the clusters of their mean dispersion [94]:

$$\mathrm{BallHall}(\mathbb{C}) = \frac{1}{K} \sum_{C_k \in \mathbb{C}} \frac{1}{n_k} \sum_{\mathbf{x} \in C_k} \|\mathbf{x}, \mathbf{c}_k\|^2 = \frac{W_{GSS}}{K} \tag{4.44}$$

where $\mathbf{c}_k$ is the centroid of cluster C_k.

4.2.5 Calinski-Harabasz Index (VRC)

Calinski-Harabasz Index [53], sometimes called the Variance Ratio Criterion (VRC), makes usage of cluster centers. Defined as:

$$\mathrm{VRC}(\mathbb{C}) = \frac{\mathrm{Tr}(\mathbf{B})}{\mathrm{Tr}(\mathbf{W})} \cdot \frac{N-K}{K-1} = \frac{B_{GSS}}{W_{GSS}} \cdot \frac{N-K}{K-1} \tag{4.45}$$

where $\mathbf{W}$ is the within-group defined in 4.26 and $\mathbf{B}$ is the between-group matrix from equation 4.19.

The trace is computed as a sum of elements on diagonal. Small and compact clusters are expected to have small values of $\mathrm{Tr}(\mathbf{W})$ and large values of $\mathrm{Tr}(\mathbf{B})$. Hence, the better the data partition the greater the value of the VRC. The normalization term $(N-K)/(K-1)$ prevents this index to increase monotonically with the number of clusters.

4.2.6 Banfeld-Raftery

This index is the weighted sum of the logarithms of the traces of the variance-covariance matrix of each cluster [42].

$$\mathrm{BR}(\mathbb{C}) = \sum_{k=1}^{K} n_k \log\left(\frac{\mathrm{Tr}\left(\mathbf{W}^{\{k\}}\right)}{n_k}\right) = \sum_{k=1}^{K} n_k \log\left(\frac{W_{GSS}^{\{k\}}}{n_k}\right) \tag{4.46}$$

When cluster contains only a single data point, the trace is equal to 0 and the logarithm is undefined.

4.2.7 Det Ratio $|T|/|W|$

The Det Ratio index [90, 95] is defined as ratio between determinant of total scatter and within-group matrices:

$$\mathrm{DetRatio}(\mathbb{C}) = \frac{\det(\boldsymbol{T})}{\det(\mathbf{W})} \tag{4.47}$$

4.2.8 Log Det Ratio

A logarithmic variant of the Det Ratio index [95]:

$$\mathrm{LDR}(\mathbb{C}) = N \log\left(\frac{\det(\boldsymbol{T})}{\det(\mathbf{W})}\right) \tag{4.48}$$

4.2.9 Friedman (TraceWiB)

An index proposed by Friedman and Rubin [90].

$$\text{fri}(\mathbb{C}) = \text{Tr}(\boldsymbol{W}^{-1}\boldsymbol{B}) \tag{4.49}$$

4.2.10 Rubin

Another index proposed by Friedman and Rubin [90].

$$\text{rub}(\mathbb{C}) = \frac{\text{Tr}(\boldsymbol{T})}{\text{Tr}(\boldsymbol{W})} \tag{4.50}$$

4.2.11 KsqDetW $k^2|W|$

The KsqDetW index [96] (also denoted as $k^2|W|$) is computed as determinant of within group scatter matrix multiplied by number of clusters squared:

$$\text{KsqDetW}(\mathbb{C}) = K^2 \det(\mathbf{W}) \tag{4.51}$$

4.2.12 Log SS Ratio

A ratio between between and withing group traces [97]:

$$\text{LogSS}(\mathbb{C}) = \log\left(\frac{B_{GSS}}{W_{GSS}}\right) \tag{4.52}$$

4.2.13 Scott-Symons

The Scott-Symons index [95] is defined as weighted sum of the logarithms of the determinants of the variance-covariance matrix of each cluster:

$$\text{Scott}(\mathbb{C}) = \sum_{k=1}^{K} n_k \log\det\left(\frac{\boldsymbol{W}^{\{k\}}}{n_k}\right) \tag{4.53}$$

4.2.14 Krzanowski-Lai index

Krzanowski and Lai proposed [98] index similar to Calinski-Harabasz.

4.2.15 C-index

The C-index was published by Hubert and Levin [99] in 1976. It is computed as

$$C_{\text{index}}(\mathbb{C}) = \frac{S_w - S_{min}}{S_{max} - S_{min}} \tag{4.54}$$

where

- S_w is the sum of the within cluster distances (equation 4.17).
- S_{min} is the sum of the N_w smallest distances between all the pairs of points in the entire dataset. There are N_t such pairs.
- S_{max} is the sum of the N_w larges distances between all the pairs of points in the entire dataset.

The index was found to exhibit excellent recovery characteristics by Milligan [100]. The minimum value across the hierarchy levels was used to indicate the optimal number of clusters [93].

4.2.16 McClain-Rao index

The McClain-Rao index was introduced in [101]. It is defined as the ratio between the mean within-cluster and between-cluster distances:

$$f_{\text{McR}}(\mathbb{C}) = \frac{S_w/N_w}{S_b/N_b} = \frac{N_b}{N_w}\frac{S_w}{S_b} \tag{4.55}$$

4.2.17 Baker-Hubert Gamma index

This index represents an adaptation of Goodman and Kruskal's Gamma statistic [102], that was originally proposed for ranking. Baker and Hubert [103] modified Γ for use in a clustering situation. Comparisons are made between all within-cluster distances and all between-cluster distances. A comparison is considered to be concordant (s^+) if a within-cluster distance is strictly less than a between-cluster distance, on the other hand we get a disconcordant (s^-) comparison when a within-cluster distance is strictly greater than a between-cluster distance. The index is computed as

$$\Gamma(\mathbb{C}) = \frac{s^+ - s^-}{s^+ + s^-} \tag{4.56}$$

Its value is between -1 and 1. Maximum values were taken to represent the correct hierarchy level [93].

4.2.18 G+ index

Using same notation as for the Baker-Hubert Γ index (Section 4.2.17), we can write the G+ index [104] as:

$$f_{g+}(\mathbb{C}) = \frac{s^-}{\frac{N_t(N_t-1)}{2}} = \frac{2s^-}{N_t(N_t-1)} \tag{4.57}$$

4.2.19 Tau index

This index is based on the Kendall's rank correlation coefficient [105] and it was reviewed by Rohlf [104] for clustering case in 1974. Still following the same notation as for Gamma and G+ index, the Kendall's τ between two vectors of length N_t is defined in statistics as the quantity:

$$\tau_a = \frac{s^+ - s^-}{\frac{N_t(N_t-1)}{2}} = \frac{2(s^+ - s^-)}{N_t(N_t-1)} \tag{4.58}$$

The τ_a does not account ties, so if a between-cluster distance and a within-cluster distance are equal, they do not enter in the numerator. In order to take ties into account, one modifies the denominator and defines the Kendall's τ_b:

$$\tau_b = \frac{s^+ - s^-}{\sqrt{(\nu_0 - \nu_1)(\nu_0 - \nu_2)}} \tag{4.59}$$

with

$$\nu_0 = \frac{N_t(N_t-1)}{2} \tag{4.60}$$

$$\nu_1 = \sum_i \frac{t_i(t_i-1)}{2} \tag{4.61}$$

$$\nu_2 = \sum_j \frac{u_j(u_j-1)}{2} \tag{4.62}$$

The first vector $\mathbf{t}$ consists of between pair distances while the second one ($\mathbf{u}$) is a binary one, having value 1 when a pair is located in the same cluster and 0 otherwise.

where

s^+ ... number of concordant pairs
s^- ... number of disconcordant pairs
t_i ... number tied values in the i^{th} group
u_j ... number tied values in the j^{th} group

Then the τ_c index is defined as:

$$f_{\text{tau}}(\mathbb{C}) = \frac{s^+ - s^-}{\sqrt{N_b N_w \frac{N_t(N_t-1)}{2}}} \tag{4.63}$$

4.2.20 Silhouette Width Criterion

Silhouette [106] combines criteria for cohesion and separation of clusters. It is computed individually for each object in dataset. Computing could be divided into following steps:

1. For the object x, calculate its average distance to all other objects in its cluster. We call this value $a(x)$.

$$a(x) = \frac{1}{|C_k| - 1} \sum_{y \in C_k, y \neq x} d(\mathbf{x}, \mathbf{y}) \tag{4.64}$$

2. For the object x and any cluster not containing this object, calculate object's average distance to all the objects in the given cluster. Find the minimum value and call it b.

$$b(x) = \min_{j, j \neq k} \left[\frac{1}{|C_j|} \sum_{y \in C_j} d(\mathbf{x}, \mathbf{y}) \right] \tag{4.65}$$

3. For the object x, the Silhouette coefficient is

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{b(\mathbf{x}), a(\mathbf{x})\}} \tag{4.66}$$

4. Repeat previous steps for all objects in cluster, to get an average cluster's value.

5. Afterwards sum cluster's Silhouettes and divide by number of clusters.

$$\text{Sil}(\mathbb{C}) = \frac{1}{|\mathbb{C}|} \sum_{k=1}^{|\mathbb{C}|} \left(\frac{1}{|C_k|} \sum_{\mathbf{x} \in C_k} s(\mathbf{x}) \right) \tag{4.67}$$

$$= \frac{1}{N} \sum_{i=1}^{N} s(\mathbf{x}_i) \tag{4.68}$$

The value of the Silhouette is in range between -1 and 1. A negative value is undesirable, because it means that the average distance within cluster is greater than minimal distance to other cluster.

The Silhouette coefficient is the default evaluation criterion in MATLAB. In order to obtain same numerical results as in MATLAB, one must use Euclidean distances without applying square root.

4.2.21 Simplified Silhouette index

The original index depends on computation of distances between all objects. The computation could be simplified for centroid based algorithms (e.g. k-means) by counting only distance to the cluster center (centroid) [107].

Instead of using average distance as defined in equation 4.64, distance to centriod is computed as:

$$a_c(\mathbf{x}) = d(x, \bar{\mathbf{x}}_c) \tag{4.69}$$

Then the index for each object is defined as:

$$s_c(\mathbf{x}) = \frac{b(\mathbf{x}) - a_c(\mathbf{x})}{\max\{b(\mathbf{x}), a_c(\mathbf{x})\}} \tag{4.70}$$

And the index averaged for the whole dataset:

$$SS = \frac{1}{N} \sum_{i=1}^{N} s_c(\mathbf{x}_i) \tag{4.71}$$

Simplified Silhouette is limited to numerical datasets only, for categorical attributes medoids could be used or the original Silhouette Width Criterion.

4.2.22 Alternate Simplified Silhouette

Another variant of simplified Silhouette redefines coefficient for individual object [107]:

$$s_a(\mathbf{x}) = \frac{b(\mathbf{x})}{a_c(\mathbf{x}) + \epsilon} \tag{4.72}$$

where ϵ is a small constant (e.g. 10^{-6} for normalized data) used to avoid division by zero when $a_c(\mathbf{x}) = 0$. The main difference should lie in non-linear behavior.

4.2.23 Dunn's index

Dunn's index [108] is another validity criterion that is based on geometrical measures of cluster compactness and separation. It is defined as:

$$\text{Dunn}(\mathbb{C}) = \min_{\substack{i,j \in \{1,\ldots,k\} \\ i \neq j}} \left(\frac{d(C_i, C_j)}{\max\limits_{l=1\ldots k} diam(C_l)} \right) \tag{4.73}$$

where $d(C_i, C_j)$ is the distance between clusters, which was originally defined as the minimum distance between pair of objects across cluster C_i and C_j (this corresponds to definition of *single-linkage* distance in hierarchical clustering), whereas $diam(C_l)$ is the maximum distance within a cluster.

If the dataset contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the cluster is expected to be small, therefore compact and separated clusters will have large value of Dunn's index.

Evaluation of large dataset with many clusters might be difficult because of considerable time complexity, also index is sensitive to noise.

4.2.24 Generalized Dunn Index (GDI)

Bezdek et al. [109] derived from Dunn's index 17 other variations by using different definitions of distance (distance between clusters is defined as *complete-linkage*, *average-linkage* etc.).

4.2.25 Davies-Bouldin index

Davies-Bouldin index [52] combines two measures, one related to dispersion and the other to the separation between different clusters. Mathematically,

$$f_{\mathrm{DB}}(\mathbb{C}) = \frac{1}{K}\sum_{i=1}^{K}\max_{i\neq j}\left(\frac{\bar{d}_i + \bar{d}_j}{d(\mathbf{c}_i, \mathbf{c}_j)}\right) \tag{4.74}$$

where $d(\mathbf{c}_i, \mathbf{c}_j)$ corresponds to the distance between the center of clusters C_i and C_j, $\bar{d}_i$ is the average within-group distance for cluster C_i.

$$\bar{d}_i = \frac{1}{|C_i|}\sum_{l=1}^{|C_i|} d(\mathbf{x}_i(l), \bar{\mathbf{x}}_i)$$

It is desirable for the clusters to have minimum possible similarity to each other, therefore we seek clustering that minimizes Davies-Bouldin index.

4.2.26 Ratkowsky-Lance index $C/\sqrt{k}$

The Ratkowsky-Lance index [110] (sometimes referred as $C/\sqrt{k}$) is computed from ratios considering between group scatter and overall variance in each dimension.

Let us denote for dimension j:

$$B_{GSS}(j) = \sum_{k=1}^{K} n_k \left(\mu_j^k - \mu_j\right)^2 \tag{4.75}$$

$$T_{SS}(j) = N \times Var(V_j) = \frac{N}{N-1} \sum_{i=1}^{N} (x_{ij} - \mu_j)^2 \tag{4.76}$$

where V_j is a column vector of dataset X, there are M such vectors, each having size N.

Then:

$$\mathrm{RL}(\mathbb{C}) = \frac{1}{M\sqrt{K}} \sum_{j=1}^{M} \frac{B_{GSS}(j)}{T_{SS}(j)} \tag{4.77}$$

Compact clusters tend to have low within-group variances which will produce high values of B_{GSS}/T_{SS}. Thus higher values will indicate better clustering.

4.2.27 Point Biserial

Point Biserial was originally proposed by Brogden [111] and later adapted by Milligan [100].

Simplified version can be expressed as:

$$\mathrm{PBS}_{\mathrm{simpl}}(\mathbb{C}) = \left(\frac{S_b}{N_b} - \frac{S_w}{N_w}\right) \cdot \frac{\sqrt{N_w \cdot N_b}}{N_w + N_b} \tag{4.78}$$

while the normalized version would account standard deviation for all distances:

$$\mathrm{PBS}(\mathbb{C}) = \left(\frac{S_b}{N_b} - \frac{S_w}{N_w}\right) \cdot \frac{\sqrt{N_w \cdot N_b}}{N_w + N_b} \cdot \frac{1}{\sigma} \tag{4.79}$$

4.2.28 SD index

The SD index was introduced in 2000 by Halkidi et al. [112] and it is based on concepts of *average clusters scattering* and *total separation of clusters* which were previously used by Rezaee et al. [113] for evaluation of fuzzy clusterings.

The average scattering is defined as:

$$Scatt(k) = \frac{1}{k} \sum_{i=1}^{k} \frac{\|\sigma(\bar{\mathbf{c}}_i)\|}{\|\sigma(X)\|} \tag{4.80}$$

where

$\|\mathbf{x}\|$	is a norm of a vector,
$\bar{\mathbf{c}}_i$	is a centroid of i-th cluster,
$\sigma(X)$	is the variance of the input dataset.

$\sigma(X) \in \mathbb{R}^m$ with m being the number of dataset dimensions. Variance for a dimension d (σ^d) is defined as:

$$\sigma^d = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^d - \bar{x}^d\right)^2$$
$$\|\sigma(X)\| = \sqrt{\sum_{i=1}^{m}(\sigma^d)^2}$$

The total separation is given by:

$$Dis(k) = \frac{D_{max}}{D_{min}}\sum_{i=1}^{k}\left(\sum_{j=1}^{k}\|\bar{\mathbf{c}}_i - \bar{\mathbf{c}}_j\|\right)^{-1} \tag{4.81}$$

where D_{max} is the maximum distance and D_{min} is the minimum distance between cluster centers ($\bar{c}_i$) and k is the number of clusters.

$$D_{max} = \max_{\substack{i,j\in\{1,\ldots,k\}\\ i\neq j}}(\|\bar{\mathbf{c}}_i - \bar{\mathbf{c}}_j\|) \tag{4.82}$$
$$D_{min} = \min_{\substack{i,j\in\{1,\ldots,k\}\\ i\neq j}}(\|\bar{\mathbf{c}}_i - \bar{\mathbf{c}}_j\|) \tag{4.83}$$

Then we can define SD validity index as follows:

$$f_{\mathrm{SD}}(\mathbb{C}) = \alpha \cdot Scat(k) + Dis(k) \tag{4.84}$$

where α should be a weighting factor equal to $Dis(c_{max})$ with c_{max} being the maximum number of clusters [112]. This makes perfect sense for fuzzy clustering (as it was proposed in [113]), however it is rather unclear how to compute c_{max} in case of crisp clustering when $c_{max} \gg k$ without running clustering with c_{max} as requested number of clusters. Nonetheless, [112] mentions that "SD proposes an optimal number of clusters almost irrespectively of c_{max}, the maximum number of clusters", thus we consider special case where $c_{max} = k$:

$$f_{\mathrm{SD}}(\mathbb{C}) = Dis(k) \cdot Scat(k) + Dis(k) \tag{4.85}$$
$$= Dis(k) \cdot (Scat(k) + 1) \tag{4.86}$$

4.2.29 S_Dbw

S_Dbw [114] is a validity index that considers also cluster density. In order to compute the index we have to define its two components.

Firstly, the average scattering is defined as:

$$Scatt(k) = \frac{1}{k}\sum_{i=1}^{k}\frac{\|\sigma(\bar{\mathbf{c}}_i)\|}{\|\sigma(X)\|} \tag{4.87}$$

where $\|\mathbf{x}\|$ is a norm of a vector,
$\bar{\mathbf{c}}_i$ is a centroid of i-th cluster,
$\sigma(X)$ is the variance of the input dataset.

$\sigma(X) \in \mathbb{R}^m$ with m being the number of dataset dimensions. Variance for a dimension d (σ^d) is defined as:

$$\sigma^d = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^d - \bar{x}^d\right)^2$$
$$\|\sigma(X)\| = \sqrt{\sum_{i=1}^{m}(\sigma^d)^2}$$

Secondly, the Inter-cluster density is defined:

$$Dens\_bw(\mathbb{C}) = \frac{1}{k\cdot(k-1)}\sum_{i=1}^{k}\left(\sum_{\substack{j=1\\ i\neq j}}^{k}\frac{\delta(x_{ij})}{\max|\delta(\mathbf{c}_i),\delta(\mathbf{c}_j)|}\right) \tag{4.88}$$

(4.89)

where k is the number of clusters and δ is computed from the neighborhood of a data point u:

$$\delta(x) = \sum_{i=1}^{N} f(x, y_i) \tag{4.90}$$

with

$$f(x,y) = \begin{cases} 0 & \text{if } d(x,y) > stdev \\ 1 & \text{otherwise} \end{cases} \tag{4.91}$$

where *stdev* is standard deviation computed over all clusters.
Then we can define S_Dbw validity index as follows:

$$f_{\text{S\_Dbw}}(\mathbb{C}) = Scat(\mathbb{C}) + Dens\_bw(\mathbb{C}) \tag{4.92}$$

4.2.30 The Ray-Turi index

This index is defined as ratio between sum of squared within cluster distances (Eq. 4.26) and the minimal clusters distance.

Let us denote the minimal squared distances between all cluster centriods:

$$D_{min} = \min_{k<k'} \Delta^2_{kk'} = \min_{k<k'} \|d(\mathbf{c}_k, \mathbf{c}_{k'})\|^2 \tag{4.93}$$

So, then we can write the Ray-Turi index as [115]:

$$f_{\mathrm{RT}}(\mathbb{C}) = \frac{1}{N} \frac{W_{GSS}}{D_{min}} \tag{4.94}$$

4.2.31 The Xie-Beni index

The Xie-Beni index [116] was originally proposed for case of fuzzy clustering, but is also applicable to a crisp clustering.

It is defined as a quotient between the mean quadratic error and the minimum of minimal squared distances between the items the clusters. The concept is very similar to the Ray-Turi index (section 4.2.30), except the denominator part. In case of Xie-Beni the minimal distance between any two items in clusters is used. This distance is usually reffered as a *single-link* distance.

$$\delta_{\mathrm{SL}}(C, C') = \min_{\substack{\mathbf{x}_i \in C \\ \mathbf{x}_j \in C'}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{4.95}$$

Then, the Xie-Beni is defined as:

$$\mathrm{XB}(\mathbb{C}) = \frac{1}{N} \frac{W_{GSS}}{\min_{k<k'} \delta_{\mathrm{SL}}(C_k, C_{k'})^2} \tag{4.96}$$

4.2.32 The PBM index

The PBM index [117] (name is an acronym constituted from initials of its authors, Pakhira, Bandyopadhyay and Maulik) is calculated using the distances between the items and their centroids and the distances between the centriod themselves.

Let us denote by D_{max} the largest distance between two cluster centroids:

$$D_{max} = \max_{k<k'} d(\mathbf{c}_k, \mathbf{c}_{k'}) \tag{4.97}$$

E_W is the sum of distances of the instances to their cluster centroid and E_T is the sum of distances of all instances to the global centroid of whole dataset:

$$E_W = \sum_{k=1}^{K} \sum_{x \in C_k} d(\mathbf{x}, \mathbf{c}_k) \tag{4.98}$$

$$E_T = \sum_{i=1}^{N} d(\mathbf{x}_i, \mathbf{g}) \tag{4.99}$$

Then the PBM index is defined as:

$$f_{\mathrm{PBM}} = \left(\frac{D_{max}}{K} \frac{E_T}{E_W} \right)^2 \tag{4.100}$$

4.2.33 Wemmert-Gancarski index

The Wemmert-Gançarski index [118] is computed from ratios between distances to centroids. For a datapoint $\mathbf{x}$ beloging to a cluster C_k the R quotient is computed as ratio between distance from this point to the center and distance to the closest centroid from remaining clusters:

$$R(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{c}_k\|}{\min\limits_{k' \neq k} \|\mathbf{x} - \mathbf{c}_k\|} \tag{4.101}$$

Then coefficient J_k is defined mean R in k$^{\mathrm{th}}$ cluster C_k:

$$J_k = \max \left\{ 0, 1 - \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} R(\mathbf{x}_i) \right\} \tag{4.102}$$

Finally the Wemmert-Gançarski index is defined as weighted mean of J_k quotients:

$$\mathrm{WG}(\mathbb{C}) = \frac{1}{N} \sum_{k=1}^{K} J_k \tag{4.103}$$

4.2.34 Overall deviation

Overall deviation is used to measure compactness of clustering. Sum of values for each cluster gives the score. If we would optimize clustering algorithm just according to this criteria the resulting clustering could have the same number of clusters as data points. Which is obviously not a useful clustering (in fact that is what we have as input data).

Table 4.4: Overview of external validation criteria, sorted alphabetically. The column **Concept** tries to capture core idea behind each criterion.

Index	Date	Ref.	Concept	Opt.	Short
AIC	1974	Sec. 4.2.1	likelihood	min	AIC
Ball-Hall	1965	Sec. 4.2.4	compactness	min	BH
Banfeld-Raftery	1993	Sec. 4.2.6	variance	max	BR
BIC	1978	Sec. 4.2.2	likelihood	min	BIC
C-index	1976	Sec. 4.2.15	$\frac{\text{compactness}}{\text{separation}}$	min	C
Calinski-Harabasz	1974	Sec. 4.2.5	variance	max	VRC
Compactness	2007	Sec. 4.2.35	compactness	min	com
Connectivity	2007	Sec. 4.2.36	connectedness	max	con
Davies-Bouldin	1979	Sec. 4.2.25	$\frac{\text{compactness}}{\text{separation}}$	min	DB
DetRatio	1967	Sec. 4.2.7	variance	min	dr
Dunn index	1974	Sec. 4.2.23	$\frac{\text{separation}}{\text{compactness}}$	max	dun
Dunn gen.	1998	Sec. 4.2.24	$\frac{\text{separation}}{\text{compactness}}$	max	GDI
Friedman	1967	Sec. 4.2.9	variance	max	fri
G+	1974	Sec. 4.2.18	disconcordant pairs	min	G+
Gamma	1976	Sec. 4.2.17	(dis)concordant pairs	max	bhg
KsqDetW	1971	Sec. 4.2.11	variance	max	ksq
Krzanowski-Lai	1988	Sec. 4.2.14	variance	max	KL
Log Det Ratio	1971	Sec. 4.2.8	variance	min	ldr
Log SS Ratio	1975	Sec. 4.2.12	variance	min	lss
McClain-Rao	1975	Sec. 4.2.16	$\frac{\text{compactness}}{\text{separation}}$	min	McR
Overall deviation	2007	Sec. 4.2.34	compactness	min	dev
PBM	2004	Sec. 4.2.32	$\frac{\text{separation}}{\text{compactness}}$	max	PBM
PointBiserial	1949	Sec. 4.2.27	separation - compactness	max	PBS
Ratkowsky-Lance	1978	Sec. 4.2.26	variance	max	RL
Ray-Turi	1999	Sec. 4.2.30	$\frac{\text{compactness}}{\text{separation}}$	min	RT
Rubin	1967	Sec. 4.2.10	variance	max	rub

Table 4.5: Second part of Table 4.4 with relative evaluation indexes overview.

Index	Date	Ref.	Concept	Opt.	Short
S_Dbw	2001	Sec. 4.2.29	$\frac{\text{compactness}}{\text{separation}}$	min	SDb
SAS	2005	Sec. 4.2.37	compactness	min	SAS
SCS	2005	Sec. 4.2.34	compactness	min	SAS
Scott-Symons	1971	Sec. 4.2.13	variance	max	sct
Silhouette	1987	Sec. 4.2.20	$\frac{\text{compactness}}{\text{separation}}$	max	sil
Silhouette (no sqrt)	1987	Sec. 4.2.20	$\frac{\text{compactness}}{\text{separation}}$	max	siq
Simpl. Silhouette	2004	Sec. 4.2.21	$\frac{\text{compactness}}{\text{separation}}$	max	ss
Simpl. Silhouette Alt.	2004	Sec. 4.2.22	$\frac{\text{compactness}}{\text{separation}}$	max	ssa
SD index	2000	Sec. 4.2.28	$\frac{\text{compactness}}{\text{separation}}$	min	SD
Tau	1974	Sec. 4.2.19	(dis)concordant pairs	max	Tau
TraceW	1965	Sec. 4.2.3	variance	max	trw
Wemmert-Gançarski	2000	Sec. 4.2.33	$\frac{\text{compactness}}{\text{separation}}$	max	WG
Xie-Beni	1991	Sec. 4.2.31	$\frac{\text{compactness}}{\text{separation}}$	min	XB

$$Dev(\mathbb{C}) = \sum_{C_k \in \mathbb{C}} \sum_{\boldsymbol{x}_i \in C_k} d(\boldsymbol{x}_i, \boldsymbol{c}_k) \tag{4.104}$$

where $\boldsymbol{c}_k$ is the centroid of cluster C_k, $d(x, y)$ is a chosen distance function[2]. Sometimes this measure is referred as *variance* criterion.

Handl and Knowles maximized overall deviation in multi-objective clustering together with measurement called *connectivity* [56].

The same criterion uses Zhao et al. [119], except in their case only cosine distance is considered. Sometimes it is called Sum of Centroid Similarities (SCS).

4.2.35 Compactness

In many cases a compact cluster is a sign of a good clustering. However the solution preferred by user could be a compromise between compactness and e.g. connectedness of clusters.

[2]e.g. Euclidean distance or any other function that suits the domain

$$Compact(\mathbb{C}) = \frac{1}{n} \cdot \sum_{i=1}^{k} |C_i| \cdot \frac{\sum_{r=1}^{|C_i|-1} \sum_{s=r+1}^{|C_i|} d_{rs}}{\frac{|C_i| \cdot (|C_i|-1)}{2}} = \frac{1}{n} \cdot \sum_{i=1}^{k} 2 \cdot \frac{\sum_{r=1}^{|C_i|-1} \sum_{s=r+1}^{|C_i|} d_{rs}}{|C_i|-1}$$

d_{rs} is distance between points r-th and s-th point in cluster C_i. Compactness measures average pairwise distance between points in the same cluster [6].

4.2.36 Connectivity

Connectivity [56] is reflecting connectedness of items in a cluster. Clusterings with low value of connectivity might have arbitrary shapes (non-spherical shapes, unlike solutions typically produced by algorithms like k-means). Connectivity evaluates the degree to which neighbouring data-points have been placed in the same cluster. It is computed as:

$$Conn(\mathbb{C}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_{ij}} \right), \tag{4.105}$$

where

$$x_{r,s} = \begin{cases} \frac{1}{j}, & \text{if } \nexists C_k : r \in C_k \wedge s \in C_k \\ 0, & \text{otherwise,} \end{cases}$$

nn_{ij} is the jth nearest neighbour of item i, N is the size of the data set and L is a parameter determining the number of neighbours that contribute to he connectivity measure.

4.2.37 Sum of Average Pairwise Similarities

Criterion $\mathcal{I}_1$ used by Zhao et al. [119]. Sums average within cluster distances using cosine distance. Originally proposed for document clustering where high dimensionality is common, thus applying cosine distance is a natural choice.

$$SAS(\mathbb{C}) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in C_k} \cos(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{4.106}$$

4.3 Summary

There are numerous cluster evaluation criteria described in the literature. This chapter attempts to map some of the most popular. While the list is definitely not complete it should give the reader some idea about various approaches to unsupervised clustering evaluation.

From practical perspective it is necessary to consider objective's computational complexity. Evaluation measures based on counting disconcordant pairs (Gamma – Section 4.2.17), G+ – Section 4.2.18 and Tau – Section 4.2.19) are the most expensive measures where evaluating clustering result has polynomial complexity.

For computing s^+ and s^- values we have to perform many comparisons:

$$N_b \times N_w = \frac{N(N-1)}{2} \times \sum_{k=1}^{K} \frac{n_k(n_k-1)}{2} \tag{4.107}$$

thus making these indexes infeasible for larger datasets, therefore these measures are not included in further experiments.

Moreover there are no results available suggesting that such expensive measure would bear more value than other measures.

Chapter 5

Cluster Ensembles

Cluster ensembles are aimed to combine different clustering techniques in order to provide more robust and stable solution across different domains [120, 121, 122]. Although a large number of clustering algorithms have been proposed in the literature (some of them are described in 2.4), the No Free Lunch theorem suggests [123] that there is no single clustering algorithm that would outperform the all the others on all datasets (clustering could be seen as an optimization problem that is attempting to find optimal set of clusters for given dataset). Each algorithm has its strengths for certain type of data, choosing appropriate algorithm for given data could be extremely difficult even for a skilled user. Automating algorithm selection could be done by ensemble techniques or by meta-learning that is discussed in Chapter 6.

The main objective of cluster ensembles is to combine multiple clusterings into one, preferably high-quality solution.

Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of N data points, where each $x_i \in X$ is represented by a vector of D attributes. A cluster ensemble is defined as $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ with M clusterings. Each base clustering π_i consists of a set of clusters $\pi_i = \{C_1^i, C_2^i, \ldots, C_{k_i}^i\}$, such that $\cup_{j=1}^{k_i} C_j^i = X$, where k_i is the number of clusters in a given ensemble member (it does not have to be the same for all members).

The problem is how to obtain a final clustering $\pi^* = \{C_1^*, C_2^*, \ldots, C_K^*\}$, where K is the number of clusters in the result and π^* summarizes the information from ensemble Π.

The process consists of two major steps. Firstly, we need to generate a set of clustering solutions and secondly we need to combine the information from these solutions. A typical result of the ensemble process is a single clustering. It has been shown in supervised ensembles that the best results are achieved when using a set of predictors whose errors are dissimilar [124]. Thus it is desirable to introduce diversity between ensemble members [125]. Using random initialization of the same algorithm (e.g. k-means) is usually not sufficient to introduce enough diversity into ensemble. The ensemble members might agree in most cases, thus the overall improvement might be insignificant.

5.1 Ensemble Generation Strategies

Many approaches have been used to initialize clustering solutions in order to create an ensemble that utilizes different cluster models and various data partitions.

- **Homogeneous ensembles** Base clusterings are created using repeated runs of a single clustering algorithm with varying sets of parameters. This is quite a popular approach, especially repeated runs of k-means with different center initialization have been used in [120, 126, 127]. When using k-means the number of clusters is typically fixed to $\lceil\sqrt{n}\rceil$, where n is the size of the dataset [128].

- **Varying *k*** Repeated runs of k-means with random initialization and k [126], a golden standard is using k in the range from 2 to $\lceil\sqrt{n}\rceil$.

- **Random subspacing/sampling** An ensemble is created from base clusterings that use different initial data. This could be achieved by projecting data onto different subspaces [129], [130] choosing different subsets of features [80], [131], or using data sampling techniques [132].

- **Heterogeneous ensembles** Diversity of solutions is introduced by applying different algorithms on the same dataset [133, 134]. The idea is that each algorithm might be better at detecting certain cluster type, thus each cluster might be build on different concept.

- **Mixed heuristics** Any combination of mentioned methods. E.g. [80] use several clustering algorithms with multiple subspaces of data.

5.1.1 Consensus Functions

Having multiple clusterings in an ensemble, one needs to come to an agreement on number of clusters and data points' assignment. Many functions have been proposed to derive a final clustering. When only one solution is considered as the result, it is usually referred as a consensus function, unlike meta clustering where the output is a set of multiple clusterings [6].

There are several approaches as to how to represent information contained in these base clusterings, some use matrices while others use graph representation. Firstly there is *label-assignment matrix* with dimensions $N \times M$ that for each data point hold label for each ensemble.

Another option is to represent associations using *pairwise similarity matrix* of size $N \times N$. That is a symmetrical matrix with zeros on diagonal that counts data point pair co-occurrences in the same cluster. E.g. for two data points (x_1, x_2) a value 2/3 would mean that in two out of three evaluated ensembles these data points were in the same cluster.

Finally, *binary cluster-association matrix* could be used, where data points are in rows and clusters from each ensemble member are in rows. Data points will be assigned 1 for clusters where they are members. For all other clusters value is set to 0.

The main consensus approaches usually fall into one of these categories:

- **Direct matching** Approaches using some voting scheme that proved to be useful in supervised learning domain [135].
- **Pairwise similarities** A pairwise similarity matrix is created and afterwards a clustering algorithm (e.g. hierarchical agglomerative clustering) is applied to group together items that were most frequently together in the same cluster in all the base clusterings [120]. The Cluster-based Similarity Partitioning Algorithm (CSPA) from Strehl and Ghosh [80] uses METIS [136] for partitioning a similarity matrix into k components.
- **Feature-based approach** The ensemble problem is formulated as categorical data clustering. For each data point an m-dimensional vector containing labels in base clusterings is created. The goal is to find a partition π^* which summarizes the information gathered from the partitions Π [137], [121], [138].
- **Graph based** Many methods use graph representation for capturing relationships between base clusterings. Strehl and Ghosh [80] also proposed the HyperGraph-Partitioning Algorithm (HGPA), where vertices correspond to data points and a hyperedge represents clusters. Another approach chooses COMUSA [139] which increases the weight of the edge for each occurrence of data pairs in the same cluster. Afterwards the nodes are sorted by the attachment score, which is defined as the ratio between the sum of the node's weights and its number of incident edges. The nodes with the highest attachment score are then used as a foundation for new clusters. This approach is relatively fast to compute, however it might fail to capture complex relationships between very diverse clusterings.

5.2 Cluster Ensemble Methods

Simple Voting Most methods falling into this category require the same number of clusters for all ensemble members (clusterings). The process starts with creating a contingency matrix $\Omega \in \mathbb{R}^{K \times K}$ from reference π_r and to-be relabeled π_t partitions. Each entry $\Omega(l, l^{'})$ that denotes number of co-occurrences between labels $l \in \pi_r$ and $l^{'} \in \pi_t$:

$$\Omega(l, l^{'}) = \sum_{n=1}^{N} \omega(x_i) \tag{5.1}$$

where

$$\omega(x_i) = \begin{cases} 1 & \text{if } C_r(x_i) = l \wedge C_t(x_i) = l^{'} \\ 0 & \text{otherwise.} \end{cases}$$

Having Ω, the label correspondence is solved by maximizing:

$$\sum_{l=1}^{K}\sum_{l'=1}^{K}\Omega(l,l')\Theta(l,l') \tag{5.2}$$

where $\Theta \in \mathbb{R}^{K\times K}$ is another matrix representing correspondence amongst labels of partitions π_r and π_t. An entry $\Theta(l,l') = 1$ if label $l \in \pi_r$ corresponds to $l' \in \pi_t$, otherwise $\Theta(l,l') = 0$.

According to Topchy et al. [140] the process is equivalent to maximum weight bipartite matching. A solution to this optimization problem can be also found using the Hungarian algorithm [141]. The algorithm has complexity $\mathrm{O}\,(n^3)$.

Incremental Voting The incremental voting scheme is build by incrementally adding ensemble members while updating statistics [142, 143]. A matrix $N \times K$ is used to accumulate frequency that a label is assigned to a data point.

Iterative Voting Consensus A feature-base method developed by Nguyen et al. [137] aims to obtain the consensus partition π^* of data points X from the label-assignment matrix. In each iteration IVC estimates the center of each cluster by computing a majority label for given set of points (cluster). After obtaining centers each data point is reassigned to the closest cluster center, based on Hamming distance between M-dimensional data points and cluster centers. The iterative process continues until there is no change in the target clustering π^*.

Clustering Aggregation The idea of clustering aggregation [126] is to minimize disagreement between ensemble members. A measure of disagreement between two clusterings $\pi_a, \pi_b \in \Pi$ with respect to two specific data points $x_i, x_j \in X$ is defined as follows:

$$d_{x_i,x_j}(\pi_a,\pi_b) = \begin{cases} 1 & \text{if } (C_a(x_i) = C_a(x_j) \wedge C_b(x_i) \neq C_b(x_j)) \vee \\ & (C_a(x_i) \neq C_a(x_j) \wedge C_b(x_i) = C_b(x_j)) \\ 0 & \text{otherwise.} \end{cases}$$

where $C_a(x_i)$ denotes that the label is assigned to data point $x_i \in X$.

Then the distance between two clusterings is defined as:

$$d_X(\pi_a,\pi_b) = \sum_{\forall(x_i,x_j)\in X} d_{x_i,x_j}(\pi_a,\pi_b) \tag{5.3}$$

The goal of aggregation clustering is to find a partition π^* that minimize distance to all ensemble members:

$$D(\pi^*) = \sum_{m=1}^{M} d_X(\pi_m,\pi^*) \tag{5.4}$$

5.3 Summary

This chapter presents several clustering ensemble approaches that has been studies in the literature. Described approaches usually manages to deliver more accurate partitions. Despite of higher computation complexity, cluster ensembles might find many applications in areas where high quality partitions are required. Also in many cases tedious parameter optimization of classical algorithms might not pay of, instead more efficient ensemble approach can be applied.

The scope of users for ensemble techniques seems to be limited as there is very little support for such algorithms in well-known data mining tools.

Chapter 6

AutoML Clustering

"Space. It seems to go on and on forever. But then you get to the end and a gorilla starts throwing barrels at you."

— Futurama, *Space Pilot 3000*

Current data mining tools are characterized by a plethora of algorithms but a lack of guidelines to select the right method according to the problem under analysis [144]. One of most difficult tasks is to predict when an algorithm is better than the other to solve a given problem [145]. Meta-learning approaches has been successfully applied supervised learning domain [146], transferring similar approach to an unsupervised domain is a challenging task.

The term meta-learning is typically used to describe automated process of obtaining knowledge about performance of algorithms and adapting itself in case that the system is presented again with the same task.

Every dataset might have very different underlying structure which is the reason why some algorithms perform better at given dataset than other. However determining the best matching structure of cluster is a challenging problem. A very simplified illustration of various input data type is shown on Figure 6.1 (inspired by [147]). The general idea is that we need to be able to describe connectedness and shape of clusters in order to be able to choose appropriate algorithm with adequate configuration.

6.1 Meta-Search

Features describing the problem are usually called *meta-features* or *meta-attributes*. Such features are either derived from an input data or relate to performance characteristic. A *meta-database* is a storage for keeping a knowledge database to learn from previous data exploration runs.

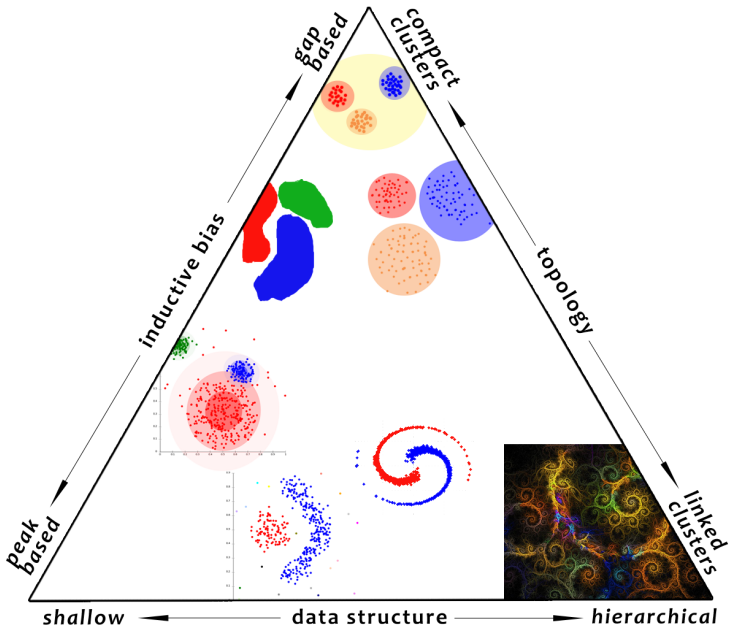


Figure 6.1: A simplified vision of clustering subspaces. For each segment different algorithm might perform better. A compact spherical cluster, on which k-means perform very well, is the easiest one to detect.

A *meta-learner* is a system that is given a set of meta-features and based on them predicts performance of given configuration (template) on data described by these features. In context of supervised learning basic statistics describing the dataset are commonly used. Examples of these statistics are: size of the dataset, number of attributes, correlation between attributes or class entropy [148, 149]. The idea is to gather descriptors about the data distribution that would correlate well with the perform of learned models [144].

Meta-features applicable to unsupervised learning can be divided into several groups. Obviously class related features frequently used in supervised learning needs to be omitted:

- *Simple descriptors*, such as input data dimensions describing basic dataset structure [150, 151, 152, 153].
- *Dimensionality reduction parameters* from methods like PCA [154].
- *Statistical features* attempting to capture data distribution, such as the kurtosis [150].
- *Landmarking features* computed from running fast machine learning algorithms [151].

Using only basic statistics proved to be insufficient for unsupervised problems. For numerical datasets the proposed meta-learning system is using following features:

- $\log_2 N$ Input data size.
- $\log_2 D$ Number of attributes.
- **AV** – Average attribute variance (σ).
- **CV** – Coefficient of variation (CV) defined as the ratio of the standard deviation σ to the attribute mean.
- **CVQ1-4** Standard deviation of all attribute's first quartiles divided by their means.
- **SKEW** – The Pearson median skewness
- **KURT** – Kurtosis (min,max, mean, std).
- **KNN4** – Average distance to 4th nearest neighbor.
- **N2ER** – Node to edge ratio after k-NN graph bisection.
- **PCA** – Basic statistics of the principal component.

Besides commonly used features the system is using descriptors computed from k-NN graph. Graph computation is quite expensive nonetheless the precomputed graph could be later reused by several algorithms. Distance to 4th neighbor was proposed Sander et al. [155] as part of a heuristic for determining optimal configuration for DBSCAN algorithm. The **KNN4** proved to have hight weight for base algorithm selection.

6.2 Measuring similarity between clusterings

One of key concepts in cluster analysis, is measuring similarity between clusterings. The lack of universally applicable cluster validity score often makes the algorithm selection and hyperparameter evaluation a tough guess [68]. Most of evaluation metrics were made on same presumptions as the algorithms which are to be validated. This task seems to be very natural, we would like to know which algorithm is giving more precise results. However creating a universal measure that would be suitable for comparing any two clusterings seems to be an impossible task.

Many different relative clustering validity criteria exist and new criteria have been proposed from time to time. And also many comparative studies between different measures has been published since 1980s. One of most extensive is work done by Milligan and Cooper [93], however there are certain conceptual flaws in used methodology. Firstly, it relies on the assumption that the accuracy of a criterion can be quantified by the number of times it indicates as the best partition. Another problem is that it relies on the assumption that a mistake made by a certain validity criterion when assessing a collection of candidate partitions of a data set can be quantified by the absolute difference between the right (known) number of clusters in the data and the number of clusters contained in the partition elected as the best one [19].

Evaluation of clustering solutions is based on criteria described in Chapter 4. However in order to obtain meaningful clusterings it is important to maintain a diversity in clustering population. Diversity is defined using Normalized Mutual Information as:

$$Diversity(\mathbb{C}, \mathbb{C}') = 1 - \text{NMI}(\mathbb{C}, \mathbb{C}') \tag{6.1}$$

which ensures that the same clustering $\mathbb{C}$ would have $Diversity(\mathbb{C}, \mathbb{C}) = 0$.

6.3 Combining Internal Criteria

There has been many studies trying to compare and select optimal evaluation measure [100, 93, 156, 157, 158, 19, 159]. Both Milligan [93] and later Shim [158] agreed that one of the top performing indices is Calinski-Harabasz index (Section 4.2.5), nonetheless both of the studies used artificially generated data that might not be enough to draw conclusions for real-world scenarios. Previous works have shown that there is no single cluster validation measure that would outperform the rest [109].

As Bezdek suggested [109] one possible approach to overcome limitations of single validity criterion is to combine multiple criteria. Surprisingly not many researchers consider combinations of clustering objectives.

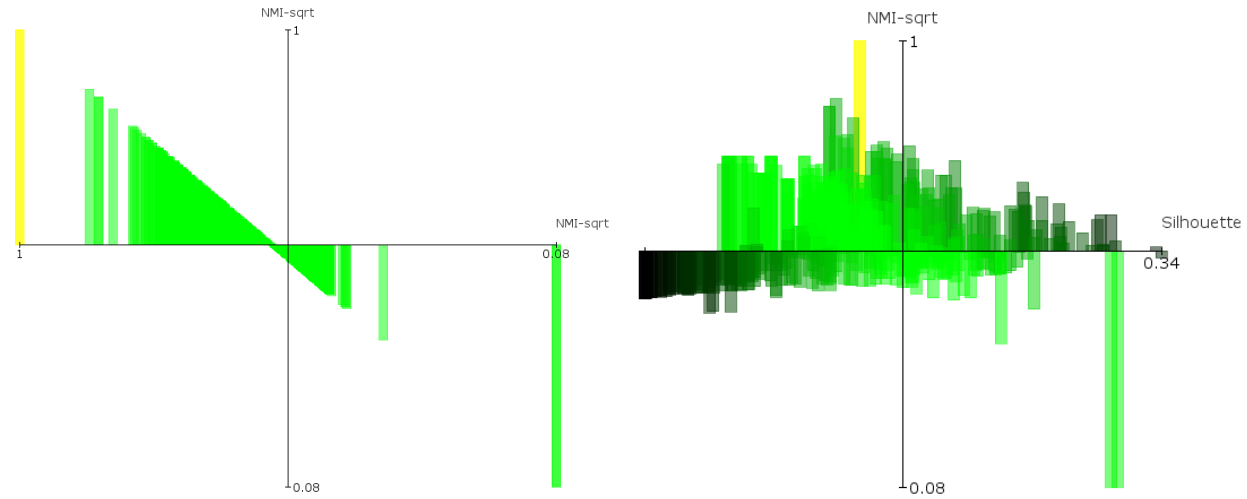
Albalate et al. [160] proposed a method based on quantiles to detect optimal algorithm, distance function and number of clusters using 5 arbitrary selected validation indexes. Best configuration is decided based on simple voting.

Jaskowiak et al. [161] consider more systematic approach into combination of clustering criteria, their work presents probably most elaborated study in the clustering criteria ensembles. The authors evaluate 28 internal validity measures, although many considered criteria are highly correlated, thus one can hardly expect improvement in overall stability and clustering quality. The study includes 18 variants of Dunn index (Section 4.2.23, 4.2.24) and 4 variants of Silhouette index (Section 4.2.20). Unlike Clustering ensembles described in Chapter 5, ensembles of validation criteria are applied in the evaluation phase.

There are many ways how could be internal clustering criteria combined together. Following sections describe combination strategies for aggregating clustering objectives.

6.3.1 Visual Comparison of Ranking Strategies

Ranking clustering results is important for ensemble approaches, evolutionary algorithms or generally for any cluster analysis where more than one result need to be compared. In any case such strategy should be based on a stable unsupervised criteria that is highly correlated with a ranking made by a human. Obtaining a human-sorted clustering ranking would be highly expensive and probably biased as well. Following experiments were conducted on already labeled data, where as the expected (human) ranking is considered ranking produced by an external criterion. Although there are multiple methods for computing



(a) Ranking by an ideal objective *(b) Ranking by Silhouette*

Figure 6.2: Plots showing ranking abilities of unsupervised evaluation measures. Each clustering is represented by one column where its height is proportional to supervised metric and its position on x axis corresponds to ranking by an unsupervised objective (best result should be placed left-most). Yellow bar represents clustering according to labels (highest NMI possible) **6.2a** Optimal ranking when the same objective is used on both axes, green color indicates high correlation between objectives. Ideally first and third quadrant should be empty. **6.2b** Clusterings ranked by Silhouette, left-most dark bars represent average solutions that are placed between top solutions while best solutions are placed somewhere in middle near yellow bar (expected clustering).

external validation mentioned in Section 4.1 the differences are subtle when compared to differences between rankings produced by various internal criteria (Section 4.2).

To demonstrate single criteria ranking abilities, series of benchmarks were performed on a set of clusterings produced by 15 clustering algorithms with various configuration. An ideal ranking criteria should be able to distinguish between good and bad results. As a reference to "correct" ranking external class labels that are not used during clustering process were used.

Figure 6.2 shows a ranking visualization. Each column represents a clustering result and height of a column is proportional to a supervised metric (Adjusted Rand Index – Section 4.1.1.2 or NMI – Section 4.1.2.1) and position on axis x corresponds to an unsupervised sorting. Best results should be leftmost placed, worst result rightmost. In an ideal case ranked clusterings would look like visualization on Figure 6.2a.

Using single (unsupervised) evaluation metric to rank a set of clusterings appears to be a hard problem, that does not seem to have a universal solution. Judging simply by number of proposed internal metrics, we can see that there does not seem to be any consensus in the clustering literature either.

Figure 6.2b shows ranking by commonly used Silhouette criterion. Left-most boxes in

black color represents clusterings considered as top results, while in reality such results are worser than a median result. While Silhouette on Iris clusterings is unable to find top results, remaining ranking is quite close to expected NMI ranking.

Ranking using Davies-Bouldin index [52] (Figure 6.4a) would place clusterings with low NMI value next to items with high NMI value. Red columns signifies incorrect placement as you can see in case of Log Det Ratio ranking on Figure 6.3c (on a green-black-red scale), clusterings ranked as top results are basically worst ones in the set. This makes Log Det Ratio or Dunn index (Figure 6.3d) unusable (at least for Iris data set) because we it can not distinguish between good and bad solutions.

Looking at Figure 6.3a we can notice between top results on left side few false positive results marked with red color, however the expected clustering (yellow bar) obtained high AIC value. It should be noted that AIC does not project all results to a single value but clusterings are almost evenly divided in the range, which is a necessary attribute of a good evaluation metric (unlike Calinski-Harabasz on Figure 6.4b where most clusterings are projected into a narrow interval).

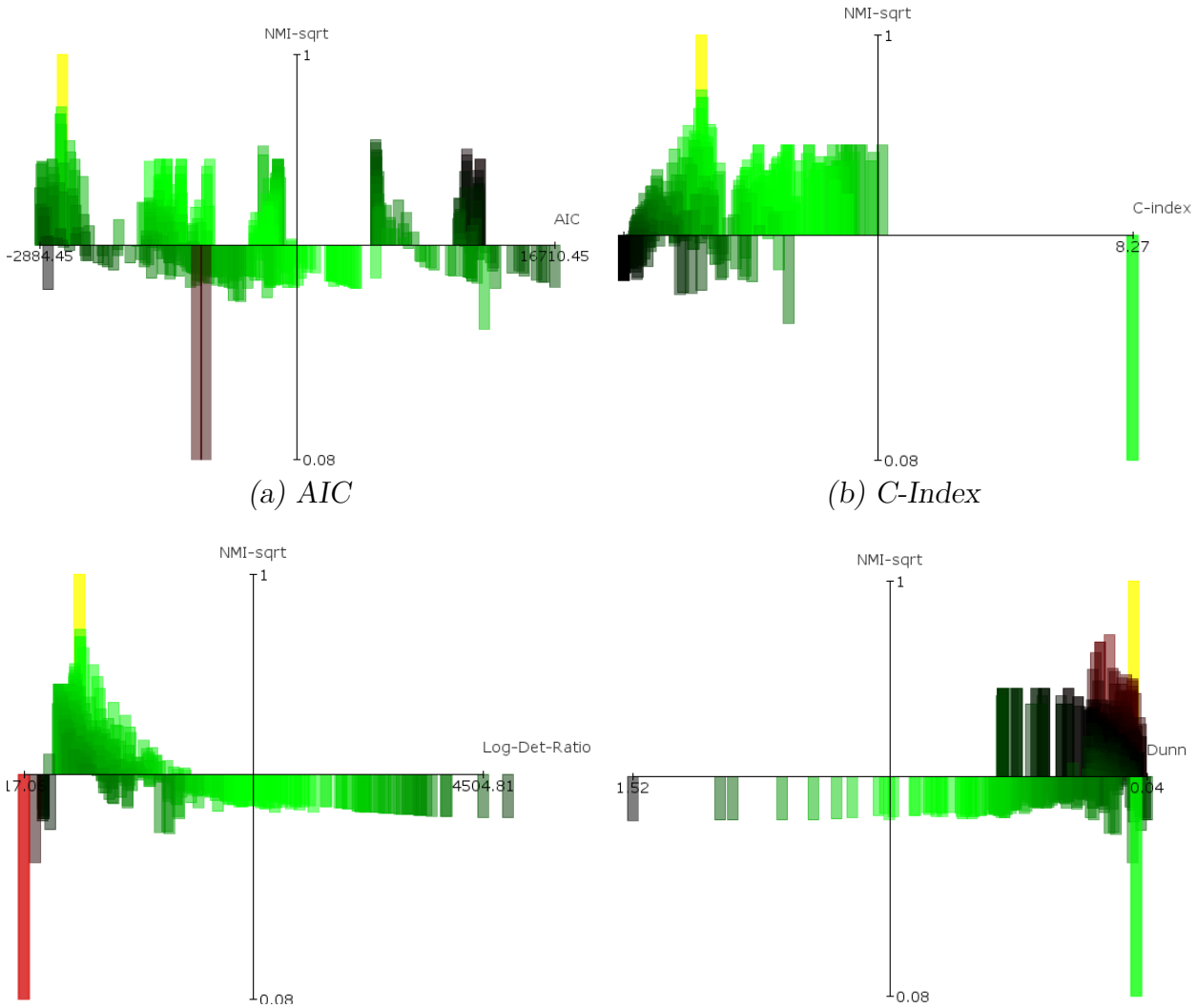
From C-index [99] visualization on Figure 6.3b we can tell that clusterings with high C-index would not correlate with high NMI values. Arguably best rankings can be found on Figures 6.4c and 6.4d where clustering were ranked by PointBiserial respective by Ratkowsky-Lance criterion. Both criteria does not rank highly expected clustering (marked with yellow bar), however excel in ranking low-quality clusterings. It would be premature to draw any hard conclusions from this observations, nonetheless the visualization could be used to empathize the differences between commonly used clustering metrics.

Comparing different relative validation indexes is complicated for many reasons. First of all, the absence of unanimous reference scale is not helping. Then, each index might be suitable for different type of data. To demonstrate properties of relative clustering validation indexes we have chosen a visualization because comparing an index with another index is not illustrative enough. After seeing such results a reader might get impression that a single objective function for ranking a set of clusterings does not exist. Apart from simple toy datasets with well separated cluster it is hard to find single criterion that correlates with external criteria.

6.3.2 Score-based Strategies

While the idea itself is fairly trivial the practical implementation requires score normalization and finding a way how to turn a minimization problem into maximization (or the other way round). This could be implemented by flipping values around their mean in cases when all score values are known in advance. Adding newly evaluated clustering to a set might require re-computation of all values. Vendramin et al. [70] proposed 4 strategies for score based combination: *Mean*, *Harmonic Mean*, *Mean-2* and *Median*.

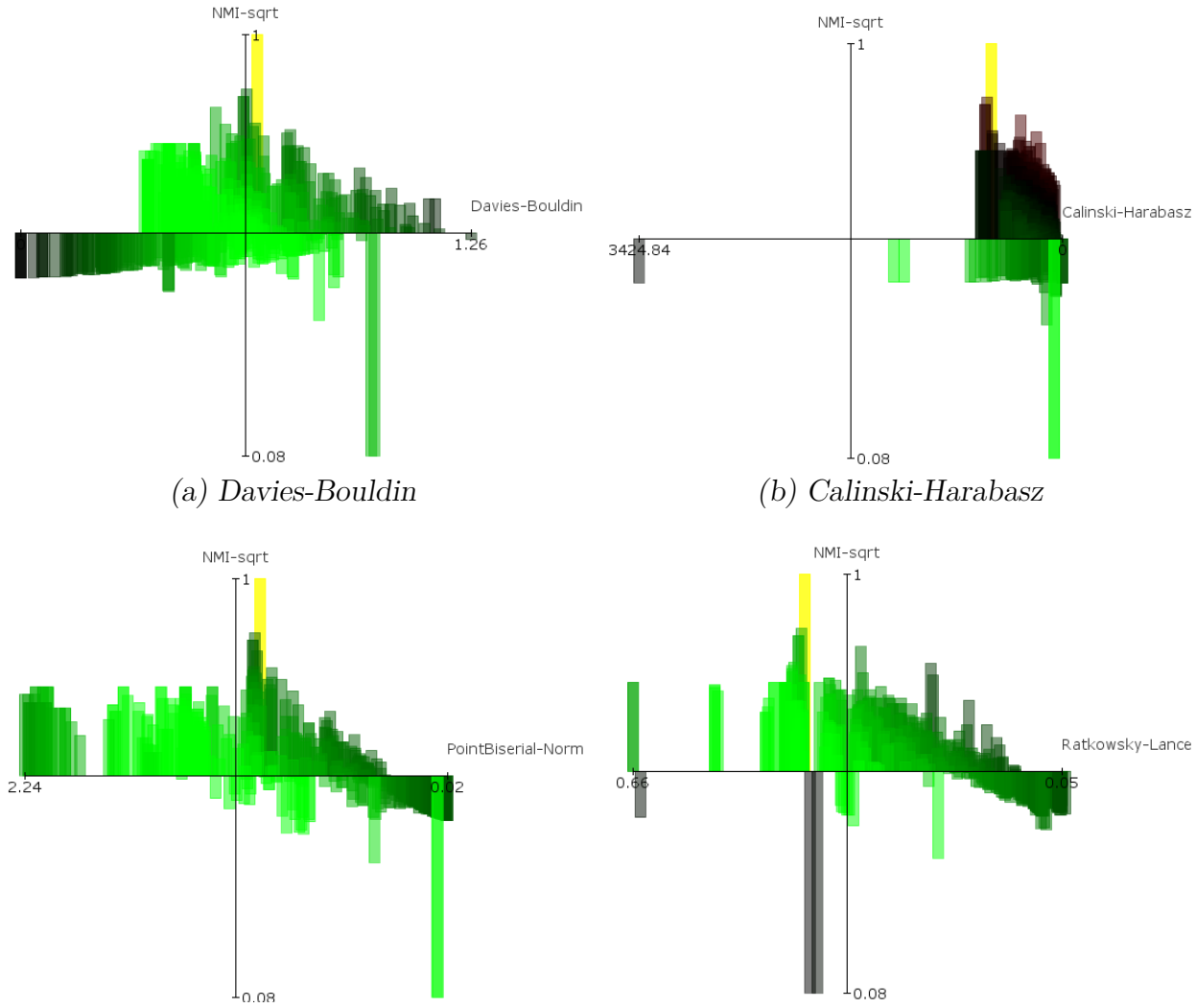
As most internal clustering criteria have unbounded definition range (meaning that the criterion might have value e.g. between 0 and ∞), all score values needs to be normalized so that all criteria contribute with the same weight. Moreover we replace all unknown values by worst known value for given criterion on predefined set of clusterings we are about to



(a) AIC *(b) C-Index*

(c) LogDet Ratio *(d) Dunn*

Figure 6.3: Visualization of Iris dataset clustering results ranking by internal validation index compared to external (supervised) validation index. Color of columns (on green-black-red scale) signifies distance from expected placement (green – correct, black – misplaced by half the scale, red – incorrect placement). Yellow column represents ideal clustering done according to external labels.



(a) Davies-Bouldin *(b) Calinski-Harabasz*

(c) PointBiserial *(d) Ratkowsky-Lance*

Figure 6.4: Visualization of Iris data set clustering results ranked by internal validation index compared to external (supervised) validation index. Color of columns (on green-black-red scale) signifies distance from expected placement (green – correct, black – misplaced by half the scale, red – incorrect placement). Yellow column represents ideal clustering done according to external labels.

evaluate. Finally the score value is scaled using Min-max normalization (Equation 2.4) to range $[1, 10]$ with best possible clustering having mean score equal to 1.

Mean Is simply computed as an arithmetic mean of all score outcomes.

Harmonic Mean Strongly penalize clusterings with at least one low score, given a score vector $\boldsymbol{s}$ of size n:

$$h_m(\boldsymbol{s}) = n \cdot \left(\sum_{i=1}^{n} \frac{1}{\boldsymbol{s}_i} \right)^{-1} \tag{6.2}$$

where n is the number of objectives we are using for evaluation. In [70] the score is demonstrated on an example where clustering with score vector $(0.00, 1.00, 0.94)$ obtains harmonic mean score 0.00. Although it is rather unclear how the operation of dividing by zero is performed, described approach appears to annul harmonic score when one criterion has 0.0 value. In the follow-up experiments each score is scaled to $[1, 10]$ so that division by zero is avoided. However after such modification the results might differ from [70] where this method provides best results.

Mean-2 Aims to remove the most discrepant value, that might be result of possibly inaccurate evaluation [70]. At least three criteria are needed in order to obtain meaningful results.

Median Sort all normalized evaluations outcomes and take the median as result. Also at least 3 criteria are needed.

6.3.3 Rank-base Strategies

Rank base strategies represent one way of avoiding a tricky score normalization, instead attempts to aggregate ordering from multiple rankings. Given a ranked list $\tau = \{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_M\}$ of M clusterings, the rank on an clustering $\mathbb{C} = \tau(i)$ is the position i of $\mathbb{C}$ in τ. Starting from 1 for the highest ranked clustering to M, for the worst clustering.

Borda count The Borda Count [162, 163] is a classical voting scheme. Given a list of M elements, the last preference should be given 1 point, the second worst option 2 points, etc. In our case the points are assigned in inversed order, but the principle remains the same. Once all lists are ranked, the final rank for each element is computed as the mean value.

Median rank Similar approach as Borda count, except instead of mean value for the final rank, median is used.

Reciprocal Rank Fusion The Reciprocal Rank fusion method (RRF) [164] uses simple formula to aggregate ranks. Given a set of ranked lists $\mathcal{T} = \{\tau_1, \ldots, \tau_t\}$, we compute RRF for an element i as:

$$RRF(\mathcal{T}(i)) = \sum_{\tau \in \mathcal{T}} \frac{1}{\epsilon + \tau(i)} \tag{6.3}$$

where ϵ is a real-valued constant. The authors suggested a value $\epsilon = 60$. The reasoning behind the constant is to eliminate outliers.

6.3.3.1 Multi-objective Ranking

Maximizing just one of these criteria would probably lead to a trivial solution (each point is a separate cluster). When using multi-objective criteria we usually want to have orthogonal measures, so that we would maximize (or minimize) properties that are contradictory.

Figure 6.5 shows an example of a Pareto front where we are optimizing two objective functions. Resulting Pareto front features equally good solutions from which we can not choose single solution that would outperform all others.

Previous section demonstrates that a single internal evaluation metrics usually does not rank clusterings as expected. Especially difficult is recognizing low quality clusterings. To overcome this issue I developed a multi-objective ranking method.

The idea behind multi-objective ranking is fairly simple: a good clustering should obtain good scores from multiple metrics. Ideally such metrics should be orthogonal. A good example of such multi-objective approach is the Chameleon algorithm mentioned earlier. However Chameleon uses a linear combination of two metrics (connectivity and separation) that can be expressed as a single number. Such algorithm can perform well on certain data type but it is hard to generalize for any data type. Also any modification to the objective function might have unpredictable behavior. Generalization of such approach turns out to be complicated.

Combining multiple existing clustering evaluation metrics is hard, as some metrics are maximized, other minimized and hardly any metric has limited range of values. Pareto optimization offers mechanism for combining objective functions where one can be maximized and other minimized. Nonetheless we end up with several Pareto fronts (similar to front on Figure 6.5) where all solutions on the same Pareto front are considered to be equal. Proposed multi-objective ranking algorithm requires at least 3 objectives (unsupervised evaluation functions) where first 2 are used for building Pareto fronts and the last one independently sorts each front.

Computing such ranking could become very expensive with continuous functions. In case of clustering metrics the situation is much simplified and many comparisons can be avoided. The method is designed for exploring large state spaces and only a top solutions are kept sorted all the time. Remaining solutions are stored as an unsorted set.

Table 6.2 shows correlation between multi-objective rankings and expected supervised NMI ranking. For most of the datasets the correlation is above 0.9 which is quite impressive

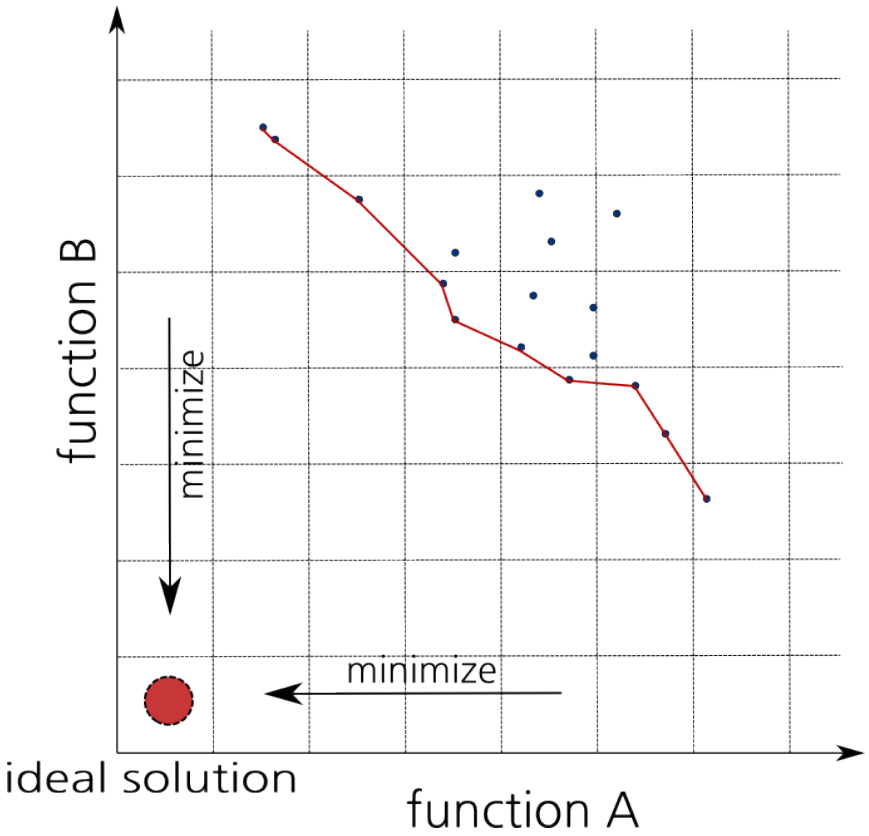
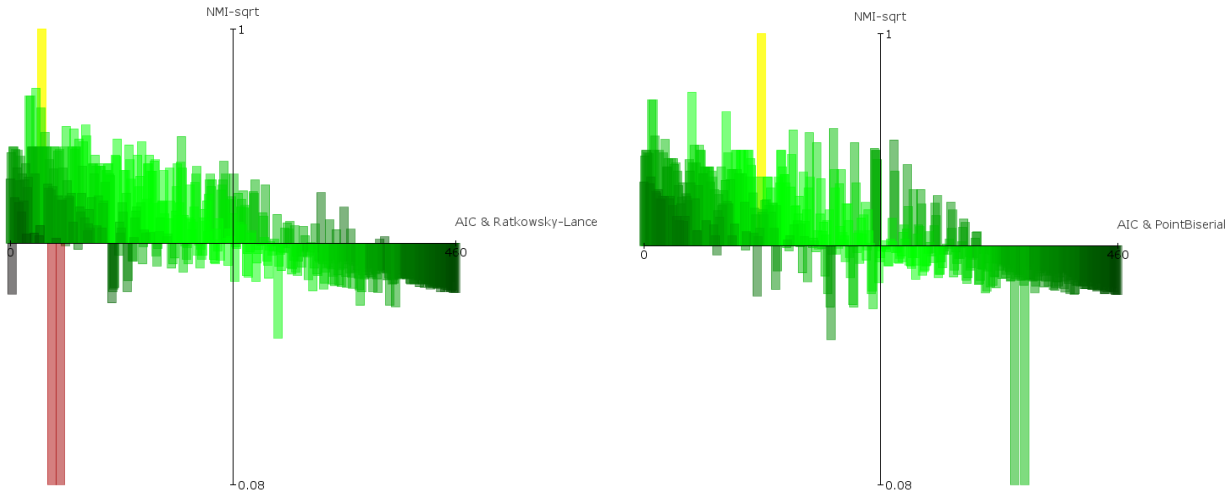


Figure 6.5: An example of a Pareto optimization. Dots represents individual solutions while on axis are shown objectives that we are trying to minimize. Typically ideal solution is unreachable, solutions connected with red line are part of the Pareto front – all of them are considered equally good.

in the area of unsupervised learning. When compared to correlations given by single objective metrics in Table 6.1, we can see that there is always a multi-objective metric that supersedes best single-objective ranking metric. The difference becomes even more obvious when set of clustering to rank contains low-quality clusterings.

Although many studies [100, 109, 19] focus on selecting optimal clustering evaluation criterion based on compactness/separation concept (see Table 4.4), a robust needs to consider multiple concepts.

AIC or BIC criteria proved to be beneficial for the multi-objective optimization and in combination with Silhouette, PointBiserial, Ratkowski-Lance can successfully navigate the search to a set of high-quality clusterings. Although it is hard to find orthogonal clustering objectives as a rule of thumb works using clustering evaluation based on completely different concepts. Like AIC – information theory based criterion and any other criterion that uses compactness (distances to cluster centroids).



(a) AIC & Ratkowski-Lance *(b) AIC & PointBiserial*

Figure 6.6: Visualization of Iris dataset clustering results ranking by a multi-objective ranking algorithm.

6.4 Comparing Ranking Strategies

Given the number of internal validation measures described before and number of ranking strategies available, there are numerous ranking configurations. While choosing optimal clustering algorithm and internal validation criteria is a difficult task for end-user, choosing optimal ranking strategy and set of internal validation criteria is infeasible.

Each evaluated dataset has class labels and expected ranking is based on external validation metric NMI_{sqrt}. Any other external criteria could be used, all ranking were evaluated against NMI_{sqrt}, ARI, VI and V-Measure. For consistency reasons only NMI_{sqrt} results are reported. Although this scenario might seem unrealistic for a real-world dataset it is essential to evaluate criterion selection on datasets with known labels before we try to transfer the knowledge to unknown datasets. For evaluation of ranking strategies was applied similar methodology to Jaskowiak et al. [161] approach. It can be summarized in a few steps:

1. Generate various clusterings using a set of clustering algorithms while fulfilling criteria for number of clusters $\lceil\sqrt{N}\rceil$.
2. Compute the internal and external validity criteria for all clusterings.
3. Compute correlation between ranking computed by external and internal criteria.
4. Repeat the process in order to verify repeatability of results.

The clustering algorithms used in experiments included k-means [20], agglomerative hierarchical clustering with several variants (single linkage, complete linkage, average linkage

and Ward's linkage), PAM [16], DBSCAN [46], CURE [39], Fast-Community [165], Chinese Whispers [166], Affinity Propagation [167] and Chameleon 2 [54]. For non-deterministic algorithms each run was repeated 5 times. Each algorithm was given certain CPU time based on input data size, thus faster algorithms were more likely to be included in results. During each run approximately 500 algorithm configurations were evaluated with randomly initialized parameters. Many clusterings were rejected due to high number of clusters or did not finish within given time limit. Resulting clusterings might were produced by unique algorithm configuration, but might include clusterings with identical partitions.

As internal validation criteria were used 39 metrics described in Section 4.2. From ranking strategies were evaluated single objective ranking, Borda count (combinations of 3 criteria), Harmonic Mean (combinations of 2 criteria), Mean (combinations of 2 criteria), Mean-2 (combinations of 3 criteria), Reciprocal Rank Fusion - RRF (combinations of 2 criteria) and Multi-Objective Ranking - MO Rank (combinations of 3 criteria). That makes altogether over 38k possible ranking configurations.

Results on *flame*, *iris*, *jain* and *zoo* dataset are shown in Table 6.3, 6.4, 6.5, 6.6, respectively. There are large deviations between repeated results reported by the same strategy. This is mostly caused by differences in quality and size during each run. Sometimes only 40 valid clusterings were found, while in other cases over 300 clusterings were evaluated.

Aggregation of correlations over different datasets is not meaningful under the No Free Lunch theorem assumption. Performance data about internal validation serve as a training set for internal criteria selection. E.g. on *iris* results in Table 6.4 we can observe frequent appearances of certain criteria: Scott-Symons (sct), Ball-Hall (BH), Banfeld-Raftery (BR) that seems to agree with single-objective correlations in Figure 6.8 for *iris* dataset. It should be noted that internal criteria ensembles performed better on all tested datasets than single criterion ranking method.

6.5 AutoML Clustering

The AutoML Clustering exploration algorithm is based on a ability to rank a population of clusterings as was described in Section 6.3. A high-level scheme of the process is shown on Figure 6.7. As in case any other meta-learning system, proper functionality requires trained predictors. The effectiveness of meta-learner increases the more data has processed. Firstly meta-features described in Section 6.1 are extracted from the input dataset.

6.5.1 Internal Metric Selection

Before starting clustering space exploration a set of evaluation metrics needs to be specified. These criteria are either given by the user or automatically predicted from meta-features. The space of possible multi-objective ranking metrics is extremely large – around 38k possible configurations. Nonetheless many configurations are highly correlated or suboptimal and can be easily avoided when predictor for ranking metrics is used. Figure 6.8 is a visual confirmation of No Free Lunch theorem [123] for internal clustering validation: there is no

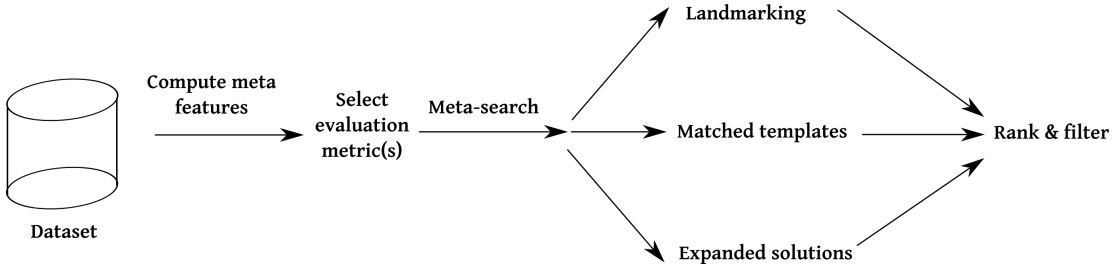


Figure 6.7: AutoML meta-search pipeline.

single criteria that would outperform all other criterion on all datasets. There are several distinguishable clusters of datasets like *chainlink*, *jain* (see Appendix A, Figure A.1d, Figure A.2d) that are based on different notion of cluster than e.g. *iris* and *ds-850* datasets. The first group has curved shapes and clusters based on connectedness in certain dimensions while the latter one contains compact and well-separated clusters. As expected for each group different set of internal validation measures performs well. Identifying meta-features that would correlate with suitability of internal metric is a crucial step towards automated clustering.

Figure 6.9 shows result of AutoML process when incorrect ranking function is used. While on Figure 6.10 ranking function includes as on of the measures Connectivity. We can see that the top-left clustering is very close to expected labeling (see Appendix A, Figure A.2b for expected Flame dataset labels). Evaluation criterion with notion of connectedness are needed for clusters that are not just round and compact.

6.5.2 Exploration

After setting up ranking, landmarking process is initiated. Landmarking runs mostly fast algorithms (e.g. k-means) and remaining of the initial clustering population is filled with clusterings produced by top-N configurations from the meta-database.

Each clustering is evaluated and meta-data such as runtime and unsupervised evaluation metrics are stored in the meta-database. Before accepting a clustering into main population several validations for ensuring correctness of clustering are performed. Also new solution needs to meet minimal *diversity-threshold* against all solutions in current population. The default threshold value is 0.1 and can be adjusted by user. This way we are making sure that we would obtain different views on the data instead of getting multiple similar clusterings produced by different algorithms.

Finally the algorithm reaches exploration phase. As in case of simulated annealing, at the beginning there is higher probability of accepting bigger changes or starting algorithm with higher complexity, meta-search also tends not to make large modification while getting closer to given limit. The parameters of each clustering might be very different. Unlike simple evolution with containing individuals with fixed binary encoded genom, in case of clustering each mutation must be done while considering parameter range for the base

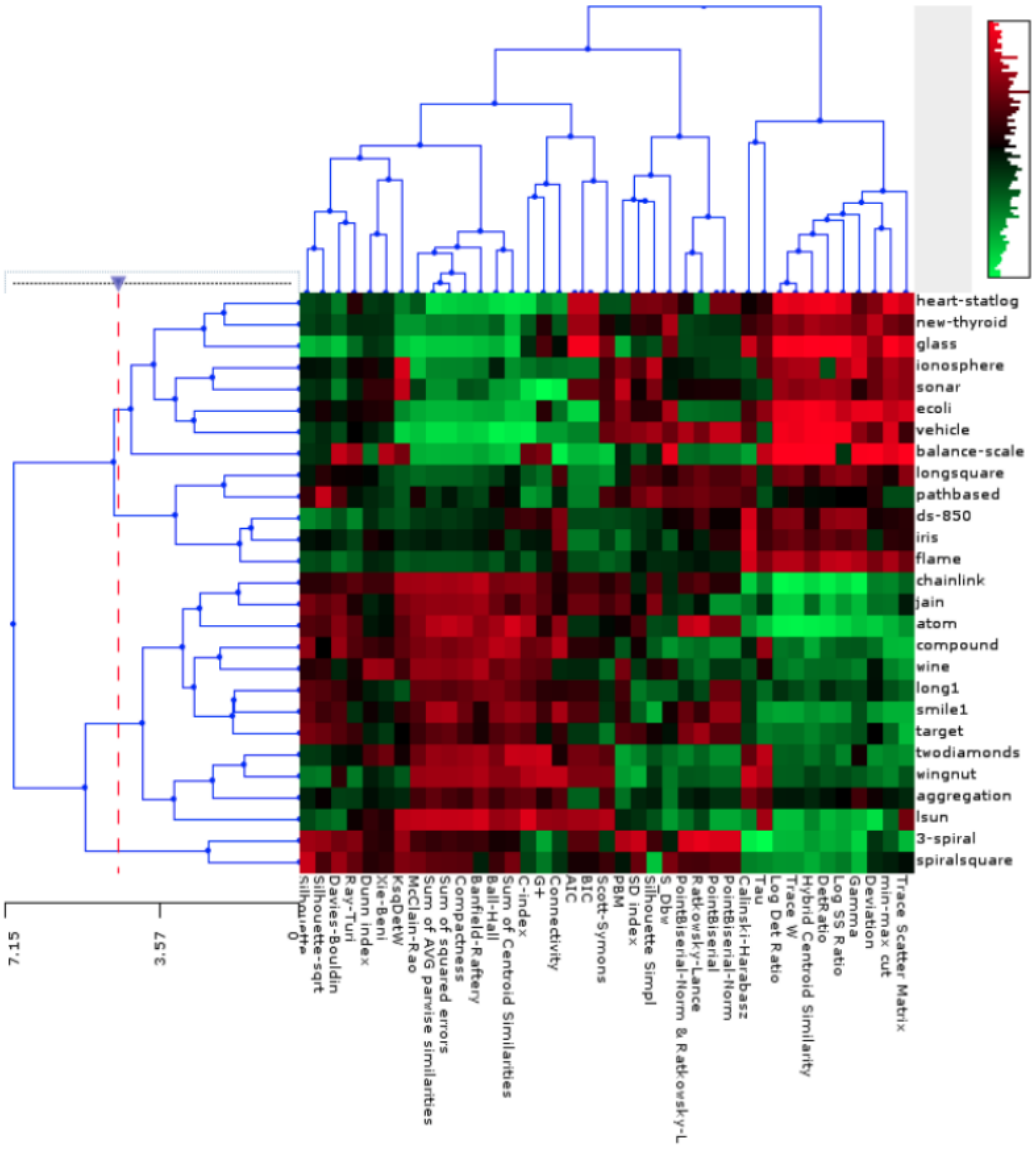


Figure 6.8: The heatmap shows ranking correlation between ranking of hierarchical clustering results by NMI and unsupervised metrics on various clustering datasets used in the literature.

Algorithm 6.1 AutoML Clustering

```
1: procedure AutomlClustering(dataset)
2:     extract meta-features
3:     choose ranking metric(s)
4:     landmarking - run fast templates
5:     find top-N templates based on meta-features
6:     rank clusterings
7:     while max. explored states not reached or time limit not reached do
8:         expand top performing templates
9:         ensemble diverse clusterings
10:        remove worst solution from population
11:    end while
12: end procedure
```

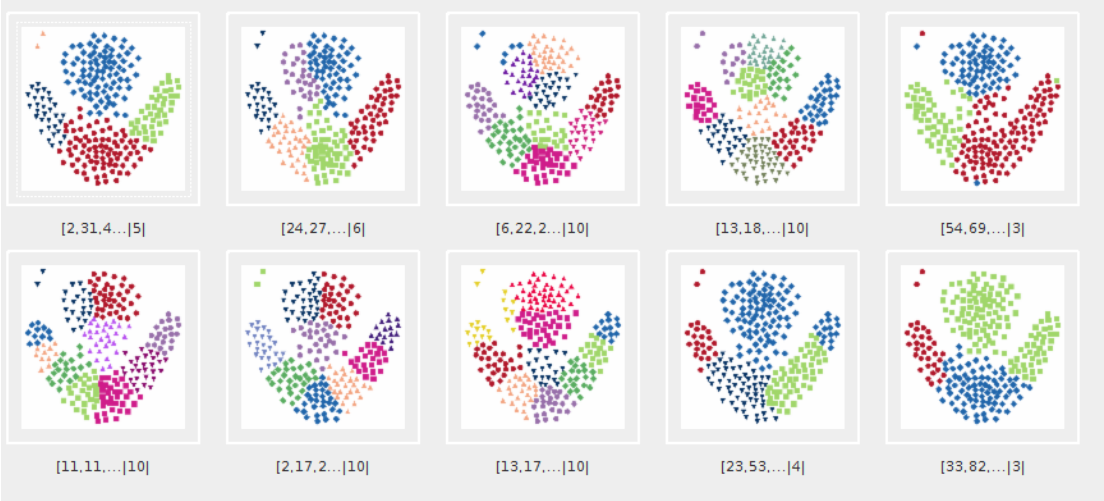


Figure 6.9: AutoML clustering using AIC and PointBiSerial for multi-objective search. Top-left clustering is supposed to be the best one, obviously this combination of objectives is not optimal.

algorithm. Thus implementing crossover for clustering algorithms would not be trivial. Currently the AutoML clustering search is mutating only single parameter at the time, as many algorithm are sensitive to even small changes and the evaluation function behaves more like in case of a discrete optimization problem. The initial version of the algorithm was inspired by NSGA-II [59].

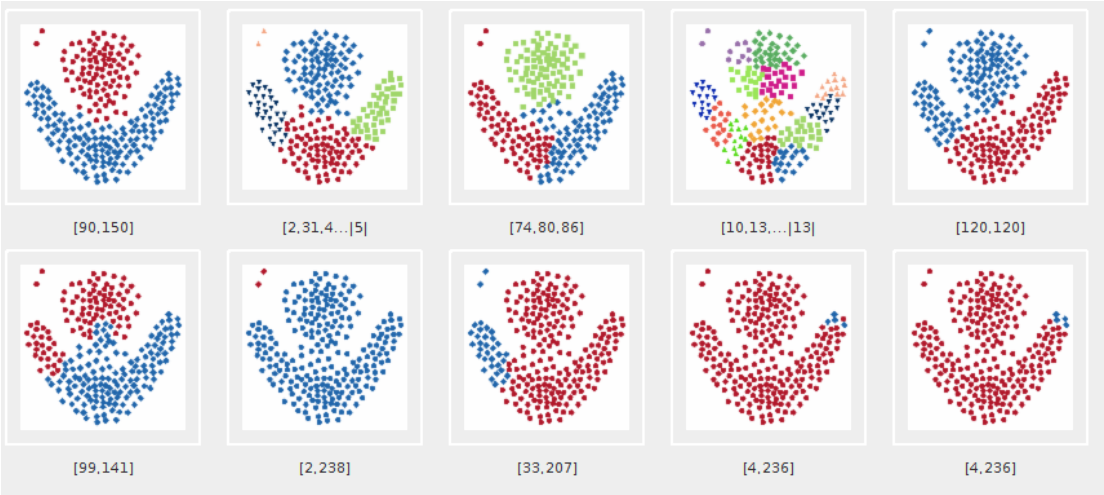


Figure 6.10: AutoML clustering using AIC and Connectivity for multi-objective search. It is apparent that Connectivity is better objective for curved-shaped clusters. The top-left solution (the best result found) is very close to human judgement (NMI = 0.91).

6.6 Summary

Implemented AutoML clustering algorithm is able to find reasonable clusterings while navigating through wast clustering space. The meta-search process is able to eliminate invalid clusterings and for many datasets produce high-quality solutions, while minimizing the number of evaluated states. Choosing the appropriate clustering is up to the end-user.

Table 6.1: Average correlation between NMI ranking and best single-objective ranking of 200 clusterings with varying quality (no limitation on number of clusters) on datasets commonly used in the literature. See Appendix A for more information about datasets.

Dataset	Best single-objective ranking	Correlation
aggregation	Deviation	0.84
atom	AIC	0.81
chainlink	PointBiserial	0.83
compound	PointBiserial	0.85
dermatology	Friedman	0.83
dpc	Dunn	0.85
ds-577	Deviation	0.85
ds-850	TrcovW	0.81
ecoli	Log-SS-Ratio	0.84
flame	SCS	0.94
glass	SAPS	0.82
ionosphere	Rubin	0.82
iris	Compactness	0.91
jain	PointBiserial-Norm	0.82
long1	PointBiserial	0.61
longsquare	AIC	0.82
lsun	TraceW	0.85
new-thyroid	Log-SS-Ratio	0.85
pathbased	Friedman	0.83
spiralsquare	TrcovW	0.69
target	TrcovW	0.84
triangle1	PointBiserial-Norm	0.82
twodiamonds	PointBiserial	0.85
wine	Ratkowsky-Lance	0.81
wingnut	PointBiserial	0.83
zelnik4	McClain-Rao	0.84
zoo	SAPS	0.81

Table 6.2: Average correlation between NMI ranking and best multi-objective ranking of 200 clusterings with varying quality. See Appendix A for more information about datasets.

Dataset	Best MO ranking	Correlation
aggregation	Compactness & TraceW & TrcovW	0.92
atom	PointBiserial & PointBiserial-Norm & AIC	0.96
chainlink	AIC & BIC & C-index	0.95
compound	Friedman & TraceSM & PointBiserial	0.90
d31	Banfeld-Raftery & Compactness & AIC	1.00
dermatology	Rubin & SSE & AIC	0.98
dpc	Silhouette-simpl & TraceSM & PointBiserial-Norm	0.85
ds-577	AIC & BIC & C-index	0.94
ds-850	AIC & BIC & AIC	0.96
ecoli	KsqDetW & Silhouette-simpl & Scott-Symons	0.97
flame	Banfeld-Raftery & SAPS & SCS	0.92
glass	PointBiserial & PointBiserial-Norm & AIC	0.97
ionosphere	Friedman & Ratkowsky-Lance & AIC	1.00
iris	Banfeld-Raftery & SSE & Connectivity	0.97
jain	DetRatio & Log-Det-Ratio & AIC	0.97
long1	Scott-Symons & TraceW & Connectivity	0.84
longsquare	McClain-Rao & TraceW & Deviation	0.93
lsun	DetRatio & Log-Det-Ratio & C-index	0.98
new-thyroid	Rubin & SSE & AIC	0.99
pathbased	Rubin & SSE & TrcovW	0.95
smile1	Rubin & SSE & AIC	0.97
sonar	Compactness & Rubin & HCS	0.82
spiralsquare	Silhouette & Silhouette-sqrt & KsqDetW	0.91
target	PointBiserial & PointBiserial-Norm & AIC	0.99
triangle1	Ball-Hall & TraceW & PointBiserial-Norm	0.98
twodiamonds	DetRatio & Log-Det-Ratio & C-index	0.96
wine	TraceW & TrcovW & AIC	0.93
wingnut	PointBiserial & PointBiserial-Norm & C-index	0.95
zelnik4	AIC & BIC & AIC	0.96
zoo	Rubin & SSE & AIC	0.96

Table 6.3: Ranking strategies comparison on *flame* dataset. Table shows best average correlations computed to a ranking computed by an external validation $\mathrm{NMI}_{\mathrm{sqrt}}$ on 10 independent runs. Each run included 40 to 240 clusterings having $k < \sqrt{N}$ found by diverse algorithms within a time limit. Table shows best performing strategies from 38k evaluated approaches. Abbreviations for internal criteria can be found in Table 4.4 and 4.5.

Method	**Objectives**	**Correlation**	$\boldsymbol{\sigma}$
MO Rank	C,PBS,SCS	0.221	0.222
MO Rank	fri,PBM,SD	0.211	0.209
MO Rank	McR,RT,siq	0.209	0.251
Mean-2	McR,ssa,WG	0.206	0.268
Mean-2	fri,RL,RT	0.206	0.235
Mean-2	C,ksq,trw	0.205	0.204
Median	dun,fri,XB	0.205	0.250
Median	RL,SAS,tsm	0.204	0.183
Median	dev,ksq,tsm	0.203	0.232
Mean-2	C,McR,XB	0.203	0.212
Median	ldr,McR,PBS	0.202	0.250
Mean-2	McR,ss,ssa	0.202	0.274
MO Rank	PBM,RL,SAS	0.202	0.230
Median	dun,fri,ldr	0.202	0.253
Harmonic Mean	VRC,McR	0.201	0.298
Median	fri,lsr,ss	0.201	0.193
Mean	C,PBS	0.201	0.214
Mean	dev,McR	0.200	0.301
Median	BR,fri,McR	0.200	0.259
Median	dev,ksq,SCS	0.200	0.254
Mean-2	C,PBM,XB	0.200	0.224
Median	fri,PBS,RL	0.200	0.238
MO Rank	fri,PBM,tcw	0.200	0.181
Mean-2	fri,McR,RL	0.199	0.245
Median	VRC,fri,lsr	0.198	0.237
Mean-2	C,dun,XB	0.198	0.190
Mean-2	C,VRC,XB	0.198	0.236
Median	lsr,McR,SSE	0.197	0.227

Table 6.4: Ranking strategies comparison on *iris* dataset. Table shows best average correlations computed to a ranking computed by an external validation NMI_{sqrt} on 10 independent runs. Each run included 60 to 300 clusterings having $k < \sqrt{N}$ found by diverse algorithms within a time limit. Table shows best performing strategies from 38k evaluated approaches. Abbreviations for internal criteria can be found in Table 4.4 and 4.5.

Method	**Objectives**	**Correlation**	$\boldsymbol{\sigma}$
MO Rank	AIC,SDb,SD	0.185	0.233
MO Rank	dun,sct,ssa	0.185	0.195
Harmonic Mean	BR,XB	0.182	0.155
Mean-2	fri,McR,sil	0.176	0.182
Mean-2	com,RL,XB	0.173	0.211
MO Rank	SDb,sct,XB	0.172	0.213
Mean	dun,fri	0.171	0.172
MO Rank	BH,SDb,ss	0.170	0.154
Mean-2	PBM,SAS,XB	0.168	0.149
Harmonic Mean	BR,PBS	0.168	0.163
Median	BH,BR,WG	0.168	0.172
Median	lsr,sil,XB	0.167	0.172
Median	BR,DB,SDb	0.166	0.196
Median	BR,SAS,WG	0.165	0.177
MO Rank	BH,SDb,ssa	0.165	0.232
Mean-2	BR,lsr,siq	0.164	0.160
Mean-2	PBM,RL,XB	0.164	0.181
MO Rank	dun,sct,SAS	0.164	0.186
Median	fri,RL,SCS	0.163	0.137
MO Rank	VRC,siq,SSE	0.163	0.149
Mean-2	VRC,RL,ssa	0.163	0.179
Median	lsr,RL,XB	0.162	0.178
Harmonic Mean	lsr,PBS	0.162	0.193
Median	McR,ss,SAS	0.160	0.175
Mean-2	DB,PBM,SSE	0.160	0.183
MO Rank	rub,SSE,tsm	0.160	0.179
MO Rank	BH,lsr,SCS	0.159	0.199
Median	BR,com,WG	0.159	0.154
Mean-2	BIC,DB,WG	0.159	0.123

Table 6.5: Ranking strategies comparison on *jain* dataset. Table shows best average correlations computed to a ranking computed by an external validation $\mathrm{NMI}_{\mathrm{sqrt}}$ on 10 independent runs. Each run included 71 to 292 clusterings having $k < \sqrt{N}$ found by diverse algorithms within a time limit. Table shows best performing strategies from 38k evaluated approaches. Abbreviations for internal criteria can be found in Table 4.4 and 4.5.

Method	**Objectives**	**Correlation**	$\boldsymbol{\sigma}$
Mean-2	sct,ssa,WG	0.347	0.252
Mean-2	BIC,tcw,WG	0.346	0.207
Mean-2	C,RL,tcw	0.340	0.221
Median	BR,SDb,SAS	0.336	0.190
Mean-2	AIC,sil,ssa	0.334	0.221
MO Rank	con,PBS,SD	0.333	0.116
Mean-2	BH,BR,SDb	0.329	0.209
Mean-2	AIC,PBM,ssa	0.329	0.260
Mean-2	SDb,siq,tsm	0.325	0.201
MO Rank	con,SD,ssa	0.324	0.204
Median	BIC,BH,ldr	0.322	0.172
Mean-2	AIC,ssa,WG	0.322	0.252
MO Rank	con,PBS,WG	0.319	0.189
Mean-2	dr,tsm,WG	0.317	0.219
Mean-2	BIC,ssa,WG	0.315	0.257
Mean	SD,trw	0.315	0.188
Mean-2	BH,SDb,sct	0.314	0.230
Median	BH,BR,dr	0.314	0.228
MO Rank	PBS,RL,tsm	0.312	0.186
Mean-2	BIC,PBM,ssa	0.312	0.248
Mean-2	BIC,BH,ldr	0.309	0.224
Median	BR,ldr,WG	0.306	0.203
Median	BR,com,SDb	0.305	0.202
Median	BH,BR,ldr	0.304	0.160
Mean-2	RL,tsm,tcw	0.304	0.182
Mean-2	BH,BR,dr	0.304	0.255
Harmonic Mean	BH,SDb	0.304	0.295
Mean-2	sct,sil,ssa	0.304	0.235
Median	dr,tsm,WG	0.303	0.188

Table 6.6: Ranking strategies comparison on *zoo* dataset. Table shows best average correlations computed to a ranking computed by an external validation NMI_{sqrt} on 10 independent runs. Each run included 60 to 200 clusterings having $k < \sqrt{N}$ found by diverse algorithms within a time limit. Table shows best performing strategies from 38k evaluated approaches. Abbreviations for internal criteria can be found in Table 4.4 and 4.5.

Method	**Objectives**	**Correlation**	$\boldsymbol{\sigma}$
MO Rank	sil,SAS,WG	0.227	0.183
MO Rank	sil,SCS,WG	0.208	0.224
MO Rank	BH,lsr,SDb	0.196	0.141
MO Rank	sil,SAS,SSE	0.193	0.202
MO Rank	BR,dun,RL	0.187	0.124
Median	ldr,lsr,WG	0.185	0.144
Mean-2	com,DB,dun	0.184	0.116
MO Rank	lsr,PBS,SD	0.182	0.171
MO Rank	BR,ss,WG	0.182	0.173
MO Rank	com,ss,WG	0.180	0.214
Mean-2	PBM,trw,WG	0.178	0.107
Median	BR,PBS,sil	0.175	0.223
Mean-2	SD,ss,ssa	0.171	0.223
Median	PBS,ss,WG	0.170	0.140
Mean-2	dev,lsr,RT	0.170	0.132
Mean-2	BIC,dun,PBS	0.170	0.229
Median	VRC,sil,SAS	0.170	0.194
MO Rank	BR,dun,SAS	0.168	0.103
Median	PBS,siq,WG	0.168	0.138
Mean-2	SDb,SD,ss	0.167	0.175
Mean-2	dr,PBS,SCS	0.167	0.216
Median	VRC,ldr,SCS	0.165	0.212
Median	VRC,com,sil	0.165	0.200
MO Rank	com,dun,ssa	0.164	0.141
Median	BH,VRC,sil	0.164	0.190
Mean-2	DB,dun,ksq	0.164	0.133
Mean-2	PBS,SD,ss	0.164	0.191
Median	dr,PBS,SCS	0.163	0.195
Median	DB,McR,XB	0.163	0.110

Chapter 7

Conclusions

7.1 Summary

This dissertation deals with advances in the automation of data-mining processes, more specifically with unsupervised clustering algorithms, their configuration and evaluation of clustering results.

7.2 Contributions of the Dissertation Thesis

This dissertation contributes to multiple areas that concerns clustering analysis.

Firstly, the Chameleon 2 algorithms push the boundaries of existing clustering algorithms with human-like clustering capabilities. Chameleon 2 uses a fine-tuned linear combination of two clustering objectives, thanks to which the algorithm is able to recognize connected clusters of various shapes.

In the area of clustering evaluation, the introduction of multi-objective ranking significantly improves the quality of discovered clusterings. Using multiple objectives turns out to be beneficial especially for low quality clusterings generated by various search methods. A visual method for inspecting ranking has also been developed.

Finally the AutoML framework has been introduced and shows promising results on clustering benchmarks. This method should simplify data clustering and will help discover an effective configuration for a given dataset.

7.3 Future Work

The author of the dissertation thesis suggests that future research should:

- Explore automated clustering methods on larger datasets with special stress on computing effectiveness.

- Consider using data subsamples for the meta-search and afterwards running full data clustering on the original dataset.
- Publish AutoML results with a concise benchmark suite.

Bibliography

[1] Jain, A. K.; Dubes, R. C. *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988, ISBN 0-13-022278-X.

[2] Boongoen, T.; Iam-On, N. Cluster ensembles: A survey of approaches with recent extensions and applications. *Computer Science Review*, volume 28, 2018: pp. 1–25.

[3] Barton, T. An Evolutionary approach for exploring clustering subspaces . Technical report, Faculty of Information Technology, Thakurova 9, 2013.

[4] Han, J.; Kamber, M. *Data mining: concepts and techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, ISBN 1-55860-489-8.

[5] Bifulco, I.; Fedullo, C.; Napolitano, F.; et al. Global optimization, meta clustering and consensus clustering for class prediction. In *Proceedings of the 2009 international joint conference on Neural Networks*, IJCNN'09, Piscataway, NJ, USA: IEEE Press, 2009, ISBN 978-1-4244-3549-4, pp. 1463–1470.

[6] Caruana, R.; Elhawary, M.; Nguyen, N.; et al. Meta Clustering. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, Washington, DC, USA: IEEE Computer Society, 2006, ISBN 0-7695-2701-9, pp. 107–118, doi:10.1109/ICDM.2006.103.

[7] Tan, P.; Steinbach, M.; Kumar, V. *Introduction to Data Mining*. Addison Wesley, first edition, May 2005, ISBN 0321321367.

[8] Jiang, D.; Tang, C.; Zhang, A. Cluster analysis for gene expression data: a survey. *IEEE Transactions on knowledge and data engineering*, volume 16, no. 11, 2004: pp. 1370–1386.

[9] Ngai, E. W.; Xiu, L.; Chau, D. C. Application of data mining techniques in customer relationship management: A literature review and classification. *Expert systems with applications*, volume 36, no. 2, 2009: pp. 2592–2602.

[10] Bhatia, S. K.; Deogun, J. S. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, volume 28, no. 3, 1998: pp. 427–436.

[11] Tao, H.; Huang, T. S. Color image edge detection using cluster analysis. In *Image Processing, 1997. Proceedings., International Conference on*, volume 1, IEEE, 1997, pp. 834–836.

[12] Henry, D. B.; Tolan, P. H.; Gorman-Smith, D. Cluster analysis in family psychology research. *Journal of Family Psychology*, volume 19, no. 1, 2005: p. 121.

[13] Kim, K.-j.; Ahn, H. A recommender system using GA K-means clustering in an online shopping market. *Expert systems with applications*, volume 34, no. 2, 2008: pp. 1200–1209.

[14] Sheppard, A. G. The sequence of factor analysis and cluster analysis: Differences in segmentation and dimensionality through the use of raw and factor scores. *Tourism Analysis*, volume 1, no. 1, 1996: pp. 49–57.

[15] Kumar, V. An Introduction to Cluster Analysis for Data Mining. February 2000. Available from: `http://www.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf`

[16] Kaufman, L.; Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. A John Wiley & Sons, Inc., 1990.

[17] Weisstein, E. W. Euclidean Metric. MathWorld – A Wolfram Web Resource, 1999. Available from: `http://mathworld.wolfram.com/EuclideanMetric.html`

[18] Jain, A. K.; Murty, M. N.; Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, volume 31, no. 3, 1999: pp. 264–323.

[19] Vendramin, L.; Campello, R.; Hruschka, E. R. On the comparison of relative clustering validity criteria. In *Proceedings of the SIAM International Conference on Data Mining, SIAM*, 2009, pp. 733–744.

[20] MacQueen, J. B. Some Methods for Classification and Analysis of MultiVariate Observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, edited by L. M. L. Cam; J. Neyman, University of California Press, 1967, pp. 281–297.

[21] Dempster, A. P.; Laird, N. M.; Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977: pp. 1–38.

[22] Lloyd, S. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, volume 28, no. 2, 1982: pp. 129–137.

[23] Manning, C.; Raghavan, P.; Schutze, H. *Introduction to information retrieval*, volume 16. Cambridge university press, 2010.

[24] Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[25] Elkan, C. Using the triangle inequality to accelerate k-means. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, 2003, p. 147.

[26] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; et al. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, volume 1, no. 7, 2002: pp. 881–892.

[27] Ng, R. T.; Han, J. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, ISBN 1-55860-153-8, pp. 144–155. Available from: `http://portal.acm.org/citation.cfm?id=645920.672827`

[28] Karypis, G.; Han, E.; Kumar, V. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *Computer*, volume 32, no. 8, August 1999: pp. 68–75, ISSN 0018-9162.

[29] Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, volume 17, no. 8, 1995: pp. 790–799.

[30] Aggarwal, C. C.; Reddy, C. K. (editors). *Data Clustering: Algorithms and Applications*. CRC Press, 2013, ISBN 978-1-46-655821-2.

[31] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, volume 43, no. 1, Jan 1982: pp. 59–69, ISSN 0340-1200, doi:10.1007/BF00337288.

[32] Furukawa, T. SOM of SOMs. *Neural Networks*, volume 22, no. 4, 2009: pp. 463–478.

[33] Tokunaga, K.; Furukawa, T. Modular network SOM. *Neural Networks*, volume 22, no. 1, 2009: pp. 82–90.

[34] Bramer, M. *Principles of Data Mining*. Springer, 2007, ISBN 1-84628-765-0.

[35] Ward, J. H. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, volume 58, no. 301, 1963: pp. 236–244.

[36] Lance, G. N.; Williams, W. T. A General Theory of Classificatory Sorting Strategies. *The Computer Journal*, volume 9, no. 4, 1967: pp. 373–380, `http://comjnl.oxfordjournals.org/content/9/4/373.full.pdf+html`.

[37] Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, volume 16, no. 1, 1973: pp. 30–34.

[38] Defays, D. An efficient algorithm for a complete link method. *The Computer Journal*, volume 20, no. 4, 1977: pp. 364–366.

[39] Guha, S.; Rastogi, R.; Shim, K. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, ACM, 1998, pp. 73–84.

[40] Figueiredo, M. A. T.; Jain, A. K. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, volume 1, no. 3, 2002: pp. 381–396.

[41] Kleinbaum, D. G.; Klein, M. Maximum likelihood techniques: An overview. In *Logistic regression*, Springer, 2010, pp. 103–127.

[42] Banfield, J. D.; Raftery, A. E. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, volume 49, 1993: pp. 803–821.

[43] Park, H.; Ozeki, T. Singularity and slow convergence of the EM algorithm for gaussian mixtures. *Neural processing letters*, volume 29, no. 1, 2009: pp. 45–59.

[44] (ed. Nong Ye), J. G. *Handbook of Data Mining*, chapter 10. Lawrence Ealbaum Assoc., 2003, pp. 247–277.

[45] Jarvis, R. A.; Patrick, E. A. Clustering using a similarity measure based on shared near neighbors. *Computers, IEEE Transactions on*, volume 100, no. 11, 1973: pp. 1025–1034.

[46] Ester, M.; Kriegel, H.; Sander, J.; et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, edited by E. Simoudis; J. Han; U. M. Fayyad, AAAI Press, 1996, ISBN 1-57735-004-9, pp. 226–231.

[47] Gan, J.; Tao, Y. DBSCAN revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, 2015, pp. 519–530.

[48] Campello, R. J.; Moulavi, D.; Sander, J. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2013, pp. 160–172.

[49] Campello, R. J.; Moulavi, D.; Zimek, A.; et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, volume 10, no. 1, 2015: p. 5.

[50] Karypis, G.; Aggarwal, R.; Kumar, V.; et al. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. *IEEE Trans. Very Large Scale Integr. Syst.*, volume 7, no. 1, Mar. 1999: pp. 69–79, ISSN 1063-8210, doi:10.1109/92.748202.

[51] Karypis, G.; Kumar, V. Multilevel k-way Hypergraph Partitioning. In *DAC*, 1999, pp. 343–348.

[52] Davies, D. L.; Bouldin, D. W. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 1, no. 2, 1979: pp. 224–227.

[53] Caliński, T.; Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, volume 3, no. 1, 1974: pp. 1–27.

[54] Barton, T.; Bruna, T.; Kordik, P. Chameleon 2: An Improved Graph-Based Clustering Algorithm. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, volume 13, no. 1, 2019: p. 10.

[55] Handl, J.; Knowles, J. Evolutionary Multiobjective Clustering. *Parallel Problem Solving from Nature - PPSN VIII*, 2004: pp. 1081–1091.

[56] Handl, J.; Knowles, J. An evolutionary approach to multiobjective clustering. *Evolutionary Computation, IEEE Transactions on*, volume 11, no. 1, 2007: pp. 56–76.

[57] Corne, D.; Jerram, N.; Knowles, J.; et al. PESA-II: region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

[58] Faceli, K.; de Souto, M. C. P.; de Araujo, D. S. A.; et al. Multi-objective clustering ensemble for gene expression data analysis. *Neurocomputing*, volume 72, no. 13-15, 2009: pp. 2763–2774.

[59] Deb, K.; Pratap, A.; Agarwal, S.; et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, volume 6, no. 2, 2002: pp. 182–197, doi:10.1109/4235.996017.

[60] Demiriz, A.; Bennett, K. P.; Embrechts, M. J. A genetic algorithm approach for semi-supervised clustering. *International Journal of Smart Engineering System Design*, volume 4, no. 1, 2002: pp. 21–30.

[61] Wagstaff, K.; Cardie, C.; Rogers, S.; et al. Constrained k-means clustering with background knowledge. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 2001, pp. 577–584.

[62] Basu, S.; Banerjee, A.; Mooney, R. Semi-supervised clustering by seeding. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 2002, pp. 19–26.

[63] Davidson, I.; Wagstaff, K. L.; Basu, S. Measuring constraint-set utility for partitional clustering algorithms. In *European conference on principles of data mining and knowledge discovery*, Springer, 2006, pp. 115–126.

[64] Inaba, M.; Katoh, N.; Imai, H. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proceedings of the tenth annual symposium on Computational geometry*, SCG '94, New York, NY, USA: ACM, 1994, ISBN 0-89791-648-4, pp. 332–339, doi:10.1145/177424.178042.

[65] Hamerly, G. Making k-means even faster. In *SIAM International Conference on Data Mining*, 2010, pp. 130–140.

[66] Jain, A. K. Data Clustering : 50 Years Beyond K-Means. *Pattern Recognition Letters*, 2010.

[67] Moulavi, D.; Jaskowiak, P. A.; Campello, R. J. G. B.; et al. Density-Based Clustering Validation. In *SDM*, edited by M. J. Zaki; Z. Obradovic; P.-N. Tan; A. Banerjee; C. Kamath; S. Parthasarathy, SIAM, 2014, ISBN 978-1-61197-344-0, pp. 839–847.

[68] Helfmann, L.; Mollenhauer, M.; von Lindheim, J.; et al. On Hyperparameter Search in Cluster Ensembles. *arXiv:stat.ML*, volume 1803, no. 11008v1, 2018.

[69] Theodoridis, S.; Koutroumbas, K. *Pattern recognition*. Academic Press, 1999, ISBN 978-0-12-686140-2, I–XIV, 1–625 pp.

[70] Vendramin, L.; Jaskowiak, P. A.; Campello, R. J. On the combination of relative clustering validity criteria. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, ACM, 2013, pp. 4–15.

[71] Wagner, S.; Wagner, D. Comparing Clusterings: An Overview. Jan. 2007.

[72] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, volume 66, no. 336, 1971: pp. 846–850.

[73] Fowlkes, E. B.; Mallows, C. L. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, volume 1, no. 383, 1983: pp. 553–569.

[74] Hubert, L.; Arabie, P. Comparing partitions. *Journal of Classification*, volume 2, no. 1, 1985: pp. 193–218.

[75] Jaccard, P. Distribution de la flore alpine dans le Bassin des Dranses et dans quelques regions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, volume 37, 1901: pp. 241–272.

[76] Chinchor, N. MUC-4 evaluation metrics. In *Proceedings of the 4th conference on Message understanding*, Association for Computational Linguistics, 1992, pp. 22–29.

[77] Rijsbergen, C. Information retrieval. 1979. Available from: `http://www.dcs.gla.ac.uk/Keith/pdf/Chapter7.pdf`

[78] Mirkin, B. Mathematical classification and clustering. 1996.

[79] Kvålseth, T. O. Entropy and Correlation: Some Comments. *IEEE Transactions on Systems, Man, and Cybernetics*, volume 17, no. 3, 1987: pp. 517–519.

[80] Strehl, A.; Ghosh, J. Cluster Ensembles: A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal on Machine Learning Research*, volume 3, December 2002: pp. 583–617, ISSN 1533-7928.

[81] Meilă, M. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, , no. 5, 2007: pp. 873–895.

[82] Rosenberg, A.; Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[83] Dom, B. E. An information-theoretic external cluster-validity measure. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 2002, pp. 137–145.

[84] Akaike, H. Information Theory and an Extension of the Maximum Likelihood Principle. *2nd International Symposium on Information Theory*, 1973: pp. 267–281.

[85] Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, volume 19, no. 6, December 1974: pp. 716–723, ISSN 0018-9286.

[86] Hastie, T.; Tibshirani, R.; Friedman, J.; et al. *The Elements of Statistical Learning*. Dordrecht: Springer, 2009, ISBN 9780387848587 0387848584.

[87] Bozdogan, H.; Sclove, S. L. Multi-sample cluster analysis using Akaike's information criterion. *Annals of the Institute of Statistical Mathematics*, volume 36, no. 1, 1984: pp. 163–180.

[88] Schwarz, G. Estimating the dimension of a model. *Annals of Statistics*, volume 6, 1978: pp. 461–464.

[89] Edwards, A. W.; Cavalli-Sforza, L. L. A method for cluster analysis. *Biometrics*, 1965: pp. 362–375.

[90] Friedman, H. P.; Rubin, J. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, volume 62, no. 320, 1967: pp. 1159–1178.

[91] Orloci, L. An agglomerative method for classification of plant communities. *The Journal of Ecology*, 1967: pp. 193–206.

[92] Fukunaga, K.; Koontz, W. L. G. A Criterion and an Algorithm for Grouping Data. *IEEE Trans. Computers*, volume 19, no. 10, 1970: pp. 917–923.

[93] Milligan, G. W.; Cooper, M. C. An Examination Of Procedures For Determining The Number Of Clusters In A Dataset. *Psychometrika*, volume 50, no. 2, June 1985: pp. 159–179.

[94] Ball, G. H.; Hall, D. J. ISODATA, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

[95] Scott, A. J.; Symons, M. J. Clustering methods based on likelihood ratio criteria. *Biometrics*, 1971: pp. 387–397.

[96] Marriott, F. Practical problems in a method of cluster analysis. *Biometrics*, 1971: pp. 501–514.

[97] Hartigan, J. A. *Clustering algorithms*. Wiley, 1975.

[98] Krzanowski, W. J.; Lai, Y. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, 1988: pp. 23–34.

[99] Hubert, L.; Levin, J. A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, volume 83, no. 6, 1976: p. 1072.

[100] Milligan, G. W. A Monte Carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, volume 46, no. 2, 1981: pp. 187–199.

[101] McClain, J. O.; Rao, V. R. Clustisz: A program to test for the quality of clustering of a set of objects. *JMR, Journal of Marketing Research (pre-1986)*, volume 12, no. 000004, 1975: p. 456.

[102] Goodman, L. A.; Kruskal, W. H. Measures of Association for Cross Classifications. *Journal of the American Statistical Association*, volume 49, no. 268, 1954: pp. pp. 732–764, ISSN 01621459.

[103] Baker, F.; Hubert, L. A graph-theoretic approach to goodness-of-fit in complete-link hierarchical clustering. *Journal of the American Statistical Association*, 1976: pp. 870–878.

[104] Rohlf, F. J. Methods of comparing classifications. *Annual Review of Ecology and Systematics*, volume 5, 1974: pp. 101–113.

[105] Kendall, M. G. A New Measure of Rank Correlation. *Biometrika*, volume 30, no. 1/2, 1938: pp. pp. 81–93, ISSN 00063444.

[106] Rousseeuw, P. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, volume 20, no. 1, 1987: pp. 53–65, ISSN 0377-0427, doi:10.1016/0377-0427(87)90125-7.

[107] Hruschka, Eduardo R and de Castro, Leandro Nunes and Campello, Ricardo JGB. Evolutionary algorithms for clustering gene-expression data. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, IEEE, 2004, pp. 403–406.

[108] Dunn, J. C. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, volume 4, no. 1, 1974: pp. 95–104.

[109] Bezdek, J. C.; Pal, N. R. Some new indexes of cluster validity. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, volume 28, no. 3, 1998: pp. 301–315.

[110] Ratkowsky, D.; Lance, G. A criterion for determining the number of groups in a classification. *Australian Computer Journal*, volume 10, no. 3, 1978: pp. 115–117.

[111] Brogden, H. E. A new coefficient: Application to biserial correlation and to estimation of selective efficiency. *Psychometrika*, volume 14, no. 3, 1949: pp. 169–182.

[112] Halkidi, M.; Vazirgiannis, M.; Batistakis, Y. Quality Scheme Assessment in the Clustering Process. In *PKDD*, *Lecture Notes in Computer Science*, volume 1910, edited by D. A. Zighed; H. J. Komorowski; J. M. Zytkow, Springer, 2000, ISBN 3-540-41066-X, pp. 265–276.

[113] Rezaee, M. R.; Lelieveldt, B.; Reiber, J. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, volume 19, no. 3-4, 1998: pp. 237–246, ISSN 0167-8655, doi:10.1016/S0167-8655(97)00168-2.

[114] Halkidi, M.; Vazirgiannis, M. Clustering Validity Assessment: Finding the Optimal Partitioning of a Data Set. In *ICDM*, edited by N. Cercone; T. Y. Lin; X. Wu, IEEE Computer Society, 2001, ISBN 0-7695-1119-8, pp. 187–194.

[115] Ray, S.; Turi, R. H. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, India, 1999, pp. 137–143.

[116] Xie, X. L.; Beni, G. A Validity Measure for Fuzzy Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 13, no. 8, 1991: pp. 841–847.

[117] Pakhira, M. K.; Bandyopadhyay, S.; Maulik, U. Validity index for crisp and fuzzy clusters. *Pattern Recognition*, volume 37, no. 3, 2004: pp. 487–501.

[118] Wemmert, C.; Gançarski, P.; Korczak, J. J. A collaborative approach to combine multiple learning methods. *International Journal on Artificial Intelligence Tools*, volume 9, no. 01, 2000: pp. 59–78.

[119] Zhao, Y.; Karypis, G.; Fayyad, U. Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, volume 10, no. 2, 2005: pp. 141–168.

[120] Fred, A. L. N.; Jain, A. K. Combining Multiple Clusterings Using Evidence Accumulation. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 27, no. 6, 2005: pp. 835–850.

[121] Topchy, A. P.; Jain, A. K.; Punch, W. F. Clustering Ensembles: Models of Consensus and Weak Partitions. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 27, no. 12, 2005: pp. 1866–1881.

[122] Iam-On, N.; Boongoen, T.; Garrett, S. LCE: a link-based cluster ensemble method for improved gene expression data analysis. *Bioinformatics*, volume 26, no. 12, 2010: pp. 1513–1519.

[123] Wolpert, D. H.; Macready, W. G. No Free Lunch Theorem for Optimization. *IEEE Transactions on Evolutionary Computation*, volume 1, no. 1, 1997.

[124] Kittler, J.; Hatef, M.; Duin, R. P. W. Combining Classifiers. In *Proceedings of the Sixth International Conference on Pattern Recognition*, Silver Spring, MD: IEEE Computer Society Press, 1996, pp. 897–901.

[125] Kuncheva, L. I.; Vetrov, D. P. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE transactions on pattern analysis and machine intelligence*, volume 28, no. 11, 2006: pp. 1798–1808.

[126] Gionis, A.; Mannila, H.; Tsaparas, P. Clustering aggregation. *TKDD*, volume 1, no. 1, 2007.

[127] Iam-On, N.; Boongoen, T.; Garrett, S. Refining pairwise similarity matrix for cluster ensemble problem with cluster relations. In *International Conference on Discovery Science*, Springer, 2008, pp. 222–233.

[128] Iam-on, N.; Boongoen, T. Improved link-based cluster ensembles. In *IJCNN*, IEEE, 2012, ISBN 978-1-4673-1488-6, pp. 1–8.

[129] Fern, X. Z.; Brodley, C. E. Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach. In *ICML*, edited by T. Fawcett; N. Mishra, AAAI Press, 2003, ISBN 1-57735-189-4, pp. 186–193.

[130] Gullo, F.; Domeniconi, C.; Tagarelli, A. Projective clustering ensembles. *Data Min. Knowl. Discov.*, volume 26, no. 3, 2013: pp. 452–511.

[131] Yu, Z.; Wong, H.-S.; Wang, H.-Q. Graph-based consensus clustering for class discovery from gene expression data. *Bioinformatics*, volume 23, no. 21, 2007: pp. 2888–2896.

[132] Dudoit, S.; Fridlyand, J. Bagging to Improve the Accuracy of A Clustering Procedure. *Bioinformatics*, volume 19, no. 9, 2003: pp. 1090–1099.

[133] Ayad, H.; Kamel, M. Finding natural clusters using multi-clusterer combiner based on shared nearest neighbors. In *International Workshop on Multiple Classifier Systems*, Springer, 2003, pp. 166–175.

[134] Law, M. H. C.; Topchy, A. P.; Jain, A. K. Multiobjective Data Clustering. In *CVPR (2)*, 2004, pp. 424–430.

[135] Bauer, E.; Kohavi, R. *Machine learning*, volume 36, no. 1-2, 1999: pp. 105–139.

[136] Karypis, G.; Kumar, V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, volume 20, December 1998: pp. 359–392, ISSN 1064-8275, doi:10.1137/S1064827595287997.

[137] Nguyen, N.; Caruana, R. Consensus Clusterings. In *ICDM*, IEEE Computer Society, 2007, ISBN 0-7695-3018-4, pp. 607–612.

[138] Boulis, C.; Ostendorf, M. Combining Multiple Clustering Systems. In *PKDD*, *Lecture Notes in Computer Science*, volume 3202, edited by J.-F. Boulicaut; F. Esposito; F. Giannotti; D. Pedreschi, Springer, 2004, ISBN 3-540-23108-0, pp. 63–74.

[139] Mimaroglu, S.; Erdil, E. Obtaining better quality final clustering by merging a collection of clusterings. *Bioinformatics*, volume 26, no. 20, 2010: pp. 2645–2646.

[140] Topchy, A. P.; Law, M. H.; Jain, A. K.; et al. IEEE, 2004, pp. 225–232.

[141] Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, volume 5, no. 1, 1957: pp. 32–38.

[142] Dimitriadou, E.; Weingessel, A.; Hornik, K. Voting-merging: An ensemble method for clustering. In *International Conference on Artificial Neural Networks*, Springer, 2001, pp. 217–224.

[143] Ayad, H. G.; Kamel, M. S. Cluster-based cumulative ensembles. In *International Workshop on Multiple Classifier Systems*, Springer, 2005, pp. 236–245.

[144] Brazdil, P.; Carrier, C. G.; Soares, C.; et al. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[145] Kalousis, A.; Gama, J.; Hilario, M. On data and algorithms: Understanding inductive performance. *Machine learning*, volume 54, no. 3, 2004: pp. 275–312.

[146] Giraud-Carrier, C.; Vilalta, R.; Brazdil, P. Introduction to the special issue on meta-learning. *Machine learning*, volume 54, no. 3, 2004: pp. 187–193.

[147] von Luxburg, U.; Williamson, R. C.; Guyon, I. Clustering: Science or Art? In *ICML Unsupervised and Transfer Learning*, *JMLR Proceedings*, volume 27, edited by I. Guyon; G. Dror; V. Lemaire; G. W. Taylor; D. L. Silver, JMLR.org, 2012, pp. 65–80.

[148] Engels, R.; Theusinger, C. Using a Data Metric for Preprocessing Advice for Data Mining Applications. In *ECAI*, 1998, pp. 430–434.

[149] Brazdil, P. B.; Soares, C.; Da Costa, J. P. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, volume 50, no. 3, 2003: pp. 251–277.

[150] Michie, D.; Spiegelhalter, D. J.; Taylor, C.; et al. Machine learning. *Neural and Statistical Classification*, volume 13, 1994.

[151] Pfahringer, B.; Bensusan, H.; Giraud-Carrier, C. G. Meta-Learning by Landmarking Various Learning Algorithms. In *ICML*, 2000, pp. 743–750.

[152] Kalousis, A. *Algorithm selection via meta-learning*. Dissertation thesis, University of Geneva, 2002.

[153] Yogatama, D.; Mann, G. Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*, 2014, pp. 1077–1085.

[154] Bardenet, R.; Brendel, M.; Kégl, B.; et al. Collaborative hyperparameter tuning. In *International conference on machine learning*, 2013, pp. 199–207.

[155] Sander, J.; Ester, M.; Kriegel, H.; et al. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery*, volume 2, no. 2, 1998: pp. 169–194, ISSN 1384-5810, 10.1023/A:1009745219419. Available from: `http://dx.doi.org/10.1023/A:1009745219419`

[156] Weingessel, A.; Dimitriadou, E.; Dolnicar, S. An examination of indexes for determining the number of clusters in binary data sets. Technical report Working Paper 29, SFB "Adaptive Information Systems and Modeling in Economics and Management Science", 1999.

[157] Maulik, U.; Bandyopadhyay, S. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence*, , no. 12, 2002: pp. 1650–1654.

[158] Shim, Y.; Chung, J.-W.; Choi, I.-C. A Comparison Study of Cluster Validity Indices Using a Nonhierarchical Clustering Algorithm. In *CIMCA/IAWTIC*, IEEE Computer Society, 2005, ISBN 0-7695-2504-0, pp. 199–204.

[159] Vendramin, L.; Campello, R. J.; Hruschka, E. R. Relative clustering validity criteria: A comparative overview. *Statistical analysis and data mining: the ASA data science journal*, , no. 4, 2010: pp. 209–235.

[160] Albalate, A.; Suendermann, D. A combination approach to cluster validation based on statistical quantiles. In *2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*, IEEE, 2009, pp. 549–555.

[161] Jaskowiak, P. A.; Moulavi, D.; Furtado, A. C.; et al. On strategies for building effective ensembles of relative clustering validity criteria. *Knowledge and Information Systems*, volume 47, no. 2, 2016: pp. 329–354.

[162] Borda, J. Mémoire sur les élections au scrutin, Histoire de l'Académie royale des sciences pour 1781. *Paris (English Translation by De Grazia, A. 1953. Isis 44)*, 1784.

[163] Emerson, P. The original Borda count and partial voting. *Social Choice and Welfare*, volume 40, no. 2, 2013: pp. 353–358.

[164] Cormack, G. V.; Clarke, C. L.; Buettcher, S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, volume 9, 2009, pp. 758–759.

[165] Newman, M. E. Fast algorithm for detecting community structure in networks. *Physical review E*, volume 69, no. 6, 2004: p. 066133.

[166] Biemann, C. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, New York City, USA, 2006, pp. 73–80.

[167] Frey, B. J. J.; Dueck, D. Clustering by Passing Messages Between Data Points. *Science*, January 2007, ISSN 1095-9203, doi:10.1126/science.1136800.

[168] Chang, H.; Yeung, D.-Y. Robust path-based spectral clustering. *Pattern Recognition*, volume 41, no. 1, 2008: pp. 191–203.

[169] Ultsch, A.; Mörchen, F. ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. *Technical Report No. 46, Dept. of Mathematics and Computer Science, University of Marburg, Germany*, 2005.

[170] Karypis, G. Karypis Lab - CLUTO's datasets.

[171] Zahn, C. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Trans. on Computers*, volume C-20, no. 1, jan 1971: pp. 68–86.

[172] Veenman, C. J.; Reinders, M. J. T.; Backer, E. A Maximum Variance Cluster Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 24, no. 9, 2002: pp. 1273–1280.

[173] Shatovska, T.; Safonova, T.; Tarasov, I. A Modified Multilevel Approach to the Dynamic Hierarchical Clustering for Complex types of Shapes. In *ISTA*, *LNI*, volume 107, edited by H. C. Mayr; D. Karagiannis, GI, 2007, pp. 176–186.

[174] Su, M.-C.; Chou, C.-H.; Hsieh, C.-C. Fuzzy C-means algorithm with a point symmetry distance. *International Journal of Fuzzy Systems*, volume 7, no. 4, 2005: pp. 175–181.

[175] Salvador, S.; Chan, P. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. In *ICTAI*, IEEE Computer Society, 2004, ISBN 0-7695-2236-X, pp. 576–584.

[176] Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science*, volume 344, no. 6191, 2014: pp. 1492–1496.

[177] Fu, L.; Medico, E. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics*, volume 8, 2007.

[178] Jain, A. K.; Law, M. H. C. Data Clustering: A User's Dilemma. In *PReMI*, *Lecture Notes in Computer Science*, volume 3776, edited by S. K. Pal; S. Bandyopadhyay; S. Biswas, Springer, 2005, ISBN 3-540-30506-8, pp. 1–10.

[179] Fränti, P.; Virmajoki, O. Iterative shrinking method for clustering problems. *Pattern Recognition*, volume 39, no. 5, 2006: pp. 761–775.

[180] Zelnik-Manor, L.; Perona, P. Self-Tuning Spectral Clustering. In *Neural Information Processing Systems (NIPS)*, 2004, pp. 1601–1608.

[181] Bartoň, T.; Kordík, P. Evaluation of Relative Indexes for Multi-objective Clustering. In *Hybrid Artificial Intelligent Systems*, *Lecture Notes in Computer Science*, volume 9121, edited by E. Onieva; I. Santos; E. Osaba; H. Quintián; E. Corchado, Springer International Publishing, 2015, ISBN 978-3-319-19643-5, pp. 465–476, doi:10.1007/978-3-319-19644-2_39.

Reviewed Publications of the Author Relevant to the Thesis

[1] Barton, T., Bruna, T., Kordik, P. *Chameleon 2: An Improved Graph Based Clustering Algorithm.* ACM Transactions on Knowledge Discovery from Data (TKDD) 2019, 13.1: 10

[2] Barton, T., Bruna, T., Kordik, P. *MoCham: Robust Hierarchical Clustering Based on Multi-objective Optimization*, Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, p. 831–838, 2016

[3] Barton, T., Kordik, P. *Evaluation of Relative Indexes for Multi-objective Clustering.* Hybrid Artificial Intelligent Systems – Lecture Notes in Computer Science, vol. 9121, p. 465-476, ISBN: 978-3-319-19643-5, Springer International Publishing, 2015

[4] Barton, T., Kordik, P. *Using Multi-Objective Optimization for the Selection of Ensemble Members.*. CEUR Workshop Proceedings, vol. 1422, p. 108-114 ISBN: 978-1-5151-2065-0, CEUR-WS.org, 2015

[5] Barton T., Kordik, P. *Encoding Time Series Data for Better Clustering Results.* International Joint Conference CISIS‘12-ICEUTE‘12-SOCO‘12 Special Sessions Springer Berlin Heidelberg, 2013

Remaining Publications of the Author Relevant to the Thesis

[1] Barton, T. *An Evolutionary approach for exploring clustering subspaces.* Ph.D. Minimum Thesis, Faculty of Information Technology, Prague, Czech Republic, 2013.

[2] Barton T. *Cluster analysis of cell profile responses.* Master Thesis, Faculty of Electrical Engineering, Czech Technical University in Prague 2011.

Appendix A

Datasets

A.1 Artificial

Datasets used for benchmarking clustering algorithms. Intentionally contains 2D and 3D datasets with clear human-distinguishable structures in the data; some datasets are contaminated with noise to test the robustness of the algorithm (see Table A.1). All data files are available online[1].

The same datasets that were used in the Chameleon paper [28], however the datasets referred to as *DS1* and *DS2* were not available. The first one comes from CURE [39][2]. Instead of *DS2* the visually similar dataset *twodiamonds* was included. For CLUTO datasets (*t4.8k*, *t5.8k*, *t7.10k* – marked as *DS3* in [28], *t8.8k* – *DS4* in [28]), which did not contain any labels, we assigned labels based on our intuition[3]. Visualization of all datasets colored by reference labels can be found in the Supplementary material.

Another group of 6 datasets (*atom*, *chainlink*, *lsun*, *target*, *twodiamonds* and *wingnut*) comes from the FCPS[4] [169], which should serve as an elementary benchmark of clustering algorithms. The datasets were manually created in such a way that none of the traditional clustering algorithms would be able to solve them all.

On the *pathbased* dataset, which consists of an incomplete circle and two Gaussian distributed clusters inside, [168] demonstrated the limitations of standard spectral clustering. Their proposed path-based spectral clustering method provides a better result, though still not perfect.

Evaluating the quality of clustering algorithms is hard, and numerous approaches have been applied in relevant studies. While unsupervised criteria can be used during clustering [181], their performance is data dependent. The prevailing approach is to evaluate

[1] `https://github.com/deric/clustering-benchmark`

[2] Data was generated based on visual similarity to an image in the referred paper, we were unable to obtain the dataset from the original author.

[3] The Labels were added manually, the original dataset did not contain any. A visualization of the assigned labels can be found in the Supplementary material.

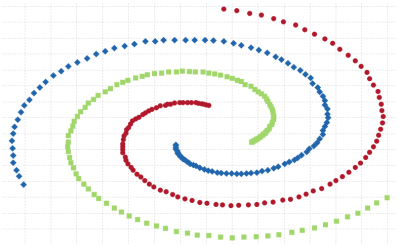
[4] Fundamental Clustering Problems Suite

Table A.1: Datasets used for our experiments: most contain patterns easily distinguishable by humans. 8 datasets contain noisy clusters.

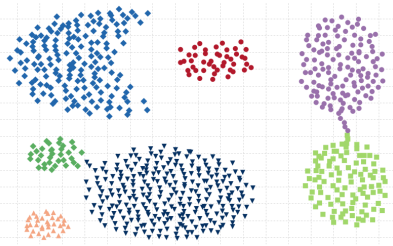
Dataset	d	n	noise size (%)	**classes**	**source**
3-spiral	2	312	-	3	[168]
aggregation	2	788	-	7	[126]
atom	3	800	-	2	[169]
chainlink	3	1000	-	2	[169]
cluto-t4.8k	2	8000	764 (9.55%)	7^3	[170]
cluto-t5.8k	2	8000	1153 (14.41%)	9^3	[170]
cluto-t7.10k	2	10000	792 (9.92%)	8^3	[170]
cluto-t8.8k	2	8000	323 (4.04%)	8^3	[170]
compound	2	399	-	6	[171]
cure-t2-4k	2	4000	200 (4.76%)	7	[39][2]
D31	2	3100	-	31	[172]
dense-disk-5k	2	5000	-	2	[173][2]
DS-850	2	850	-	5	[174]
diamond9	2	3000	-	9	[175]
disk-in-disk	2	4600	-	2	[173][2]
dpb	2	4000	657 (16.43%)	6^3	[176]
flame	2	240	-	2	[177]
impossible	2	3673	78 (2.12%)	8	[66][2]
jain	2	373	-	2	[178]
long1	2	1000	-	2	[56]
longsquare	2	900	-	6	[56]
lsun	2	400	-	3	[169]
pathbased	2	300	-	3	[168]
s-set1	2	5000	-	15^3	[179]
sizes1	2	1000	-	4	[56]
smile1	2	1000	-	4	[56]
spiralsquare	2	1500	-	6	[56]
target	2	770	-	6	[169]
twodiamonds	2	800	-	2	[169]
wingnut	2	1016	-	2	[169]
zelnik4	2	622	138 (22.19%)	5	[180]

clustering outcomes against external (ground truth) labels. In this paper, we provide labels for every dataset in the benchmark (all datasets are listed in Table A.1), therefore the quality of clustering can be evaluated using supervised criteria.

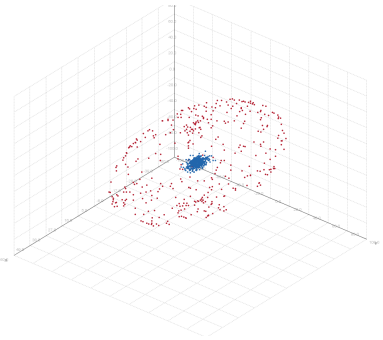
A.2 Other results



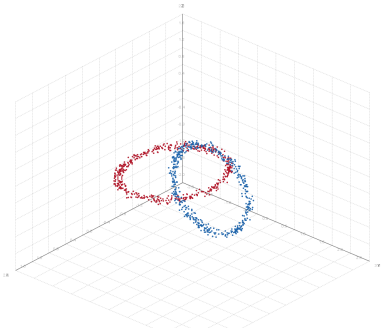
(a) Dataset *3-spiral*.



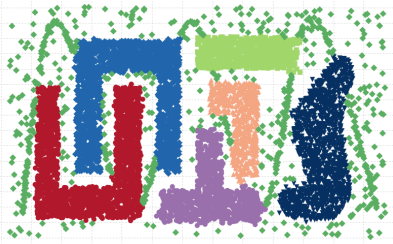
(b) Dataset *aggregation*.



(c) Dataset *atom*.

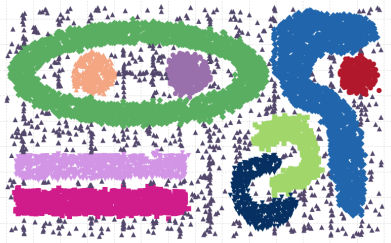


(d) Dataset *chainlink*.

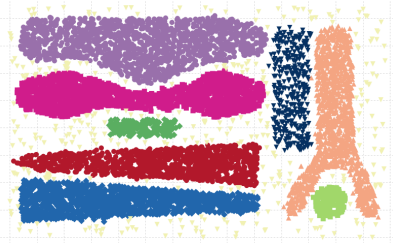


(e) Dataset *cluto-t4.8k*.

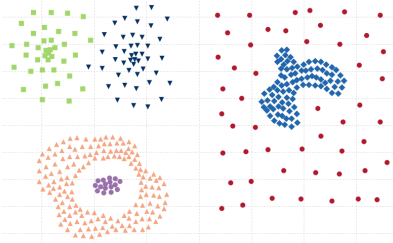
(f) Dataset *cluto-t5.8k*.



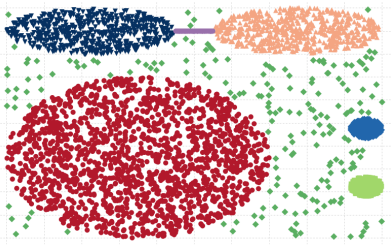
(g) Dataset *cluto-t7.10k*.



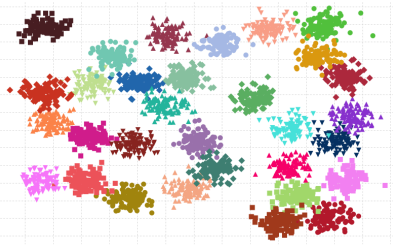
(h) Dataset *cluto-t8.8k*.



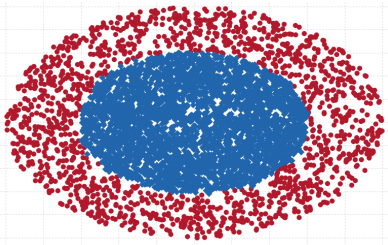
(i) Dataset *compound*.



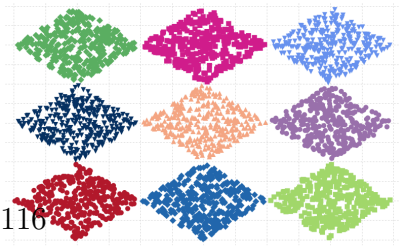
(j) Dataset *cure-t2-4k*.



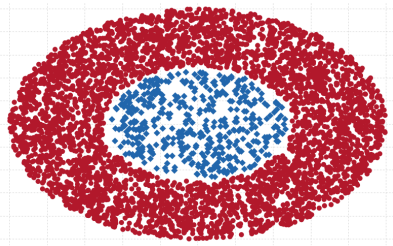
(k) Dataset *D31*.



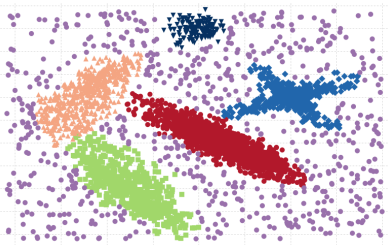
(l) Dataset *dense-disk-5k*.



(m) Dataset *diamond9*.

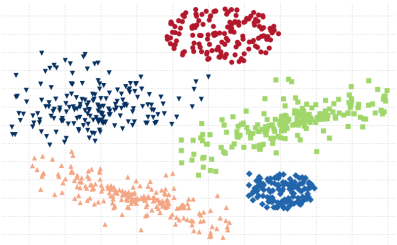


(n) Dataset *disk-in-disk*.

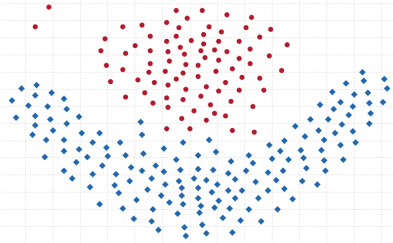


(o) Dataset *dpb*.

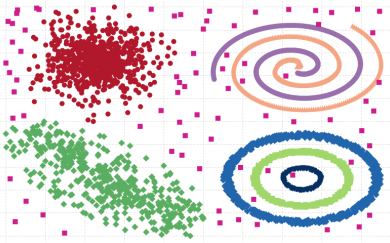
Figure A.1: Visualization of datasets used in experiments with ground truth assignments.



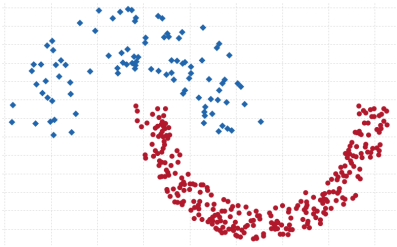
(a) Dataset *DS-850*.



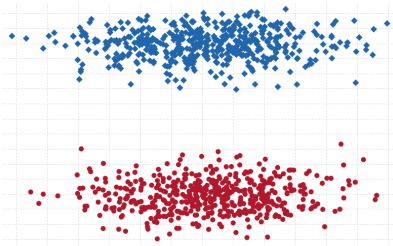
(b) Dataset *flame*.



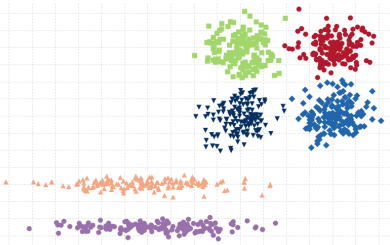
(c) Dataset *impossible*.



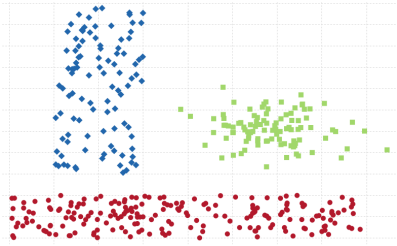
(d) Dataset *jain*.



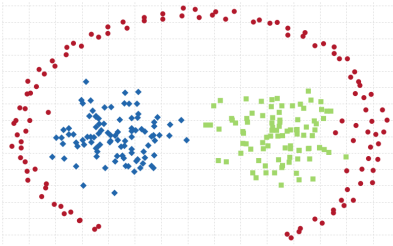
(e) Dataset *long1*.



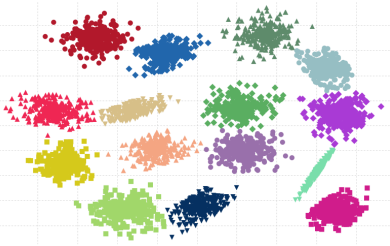
(f) Dataset *longsquare*.



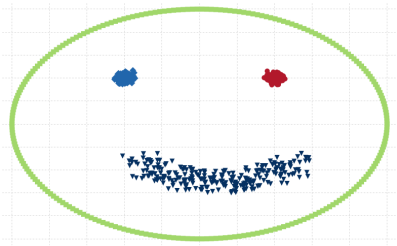
(g) Dataset *lsun*.



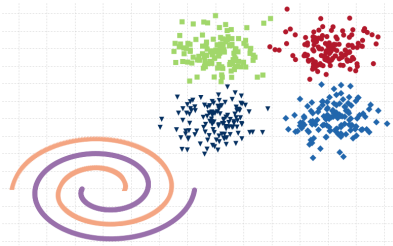
(h) Dataset *pathbased*.



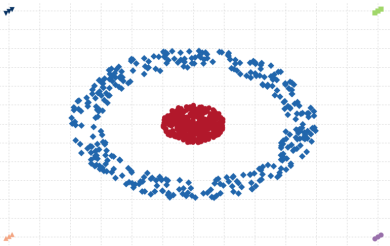
(i) Dataset *s-set1*.



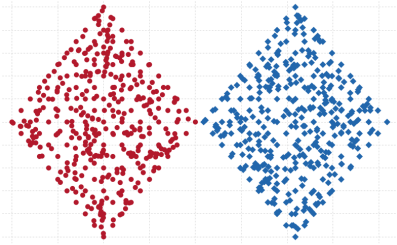
(j) Dataset *smile1*.



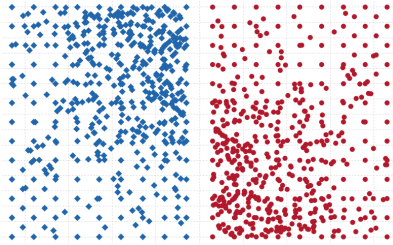
(k) Dataset *spiralsquare*.



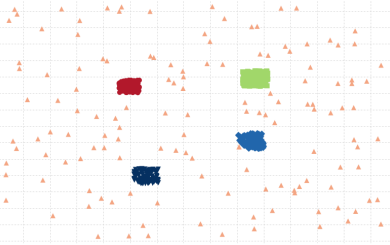
(l) Dataset *target*.



(m) Dataset *twodiamonds*.



(n) Dataset *wingnut*.



(o) Dataset *zelnik4*.

Figure A.2: Visualization of datasets used in experiments with ground truth assignments, second part.

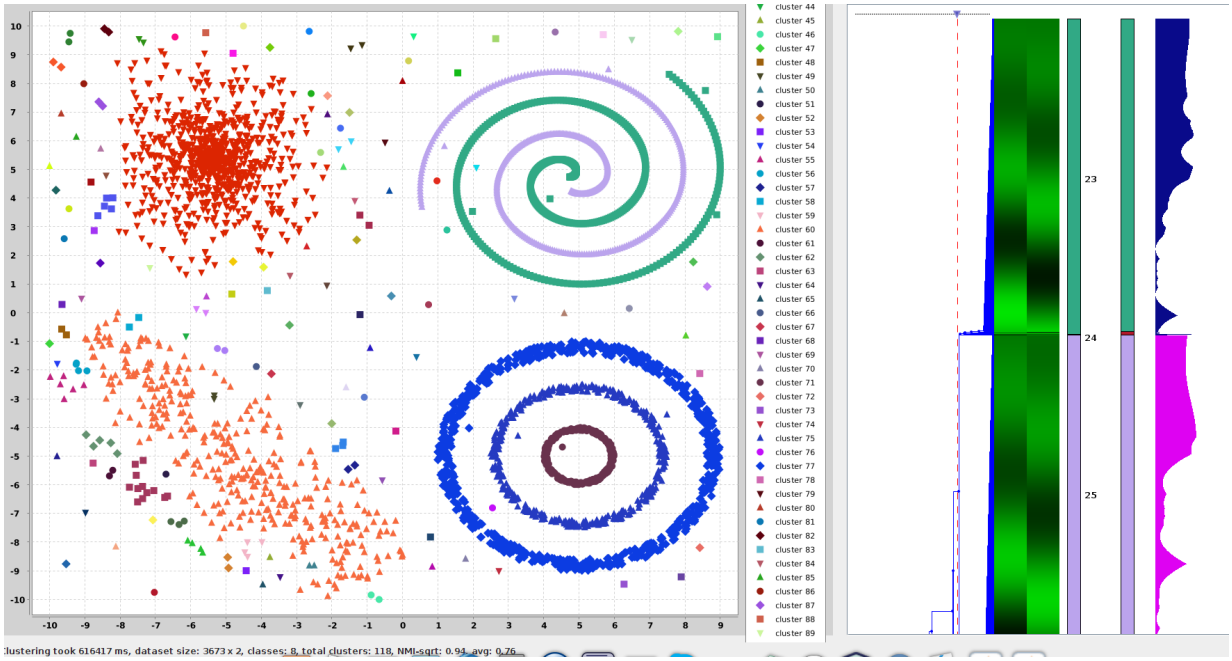


Figure A.3: Inspection tool in Clueminer displays various criteria. This particular clustering result demonstrates that the algorithm was unable to deal with noise.

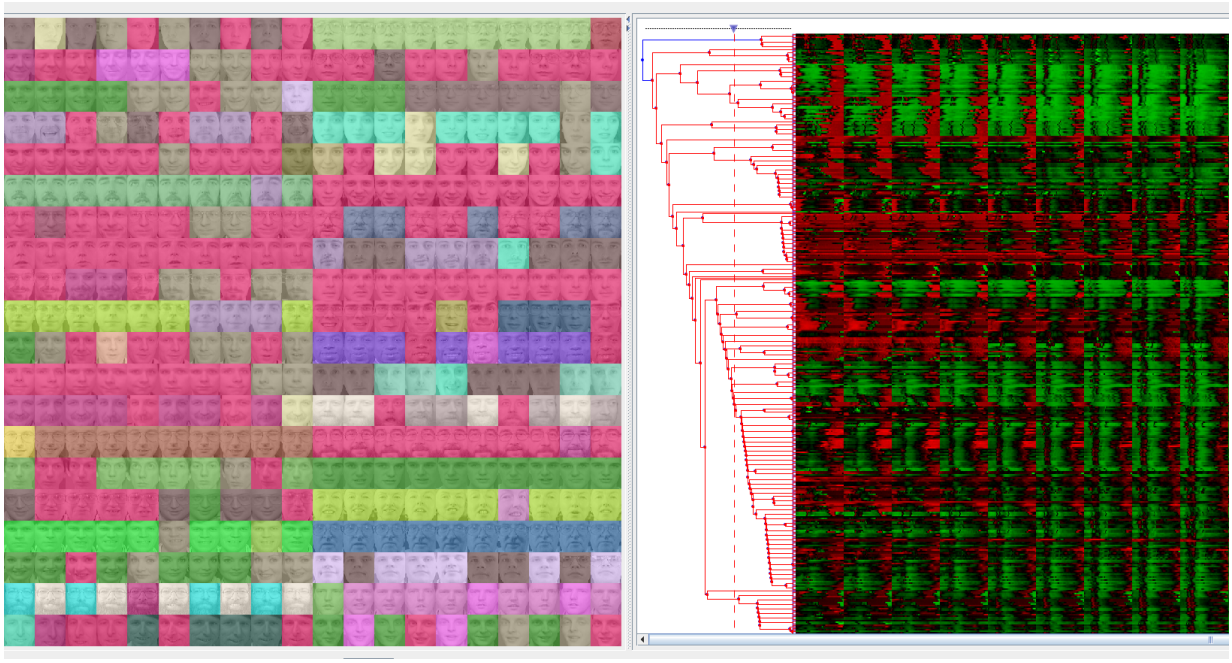


Figure A.4: We demonstrated that our Chameleon2 algorithm works well for most of low dimensional benchmarking datasets. In this particular example we show, that it is also capable to perform well on multidimensional data (grayscale vectors of face pixels).

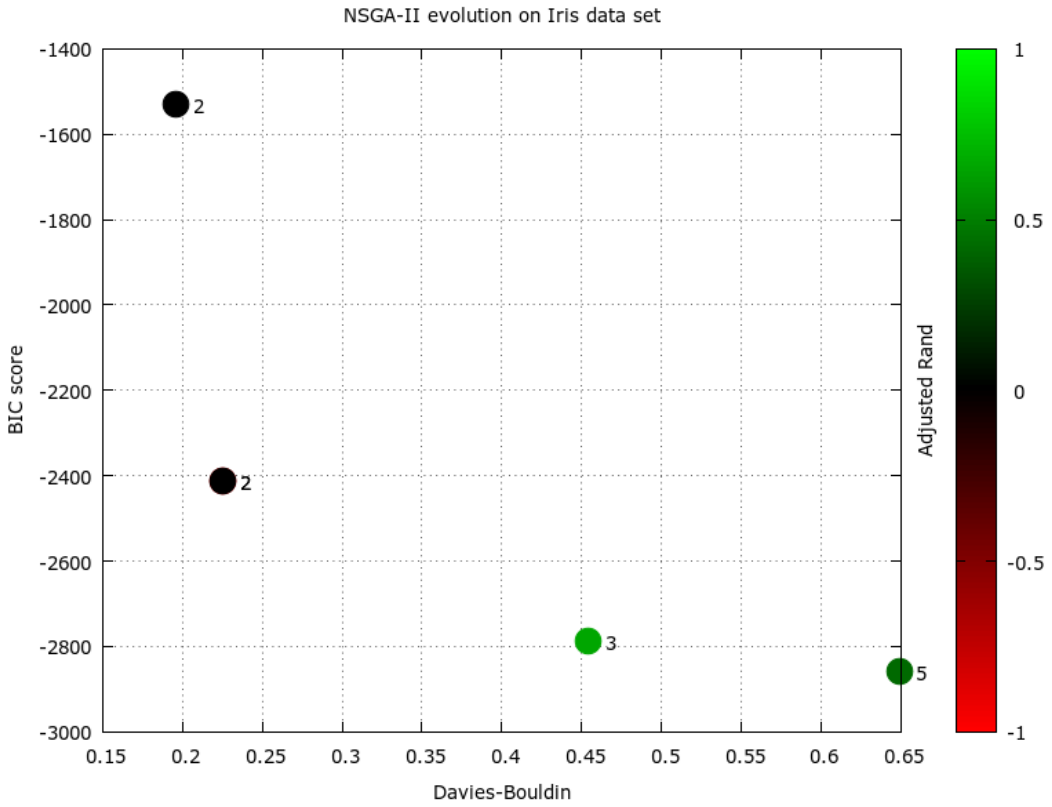


Figure A.5: When a multiobjective search is performed using NSGA2 for BIC and DB index, we are able to discover interesting clusterings on the Pareto front. Black points are clusterings with two clusters (setosa and virginica+versicolor) having quite high score for unsupervised criteria, but scoring low in ARAND index due to mixing virginica and versicolor class.

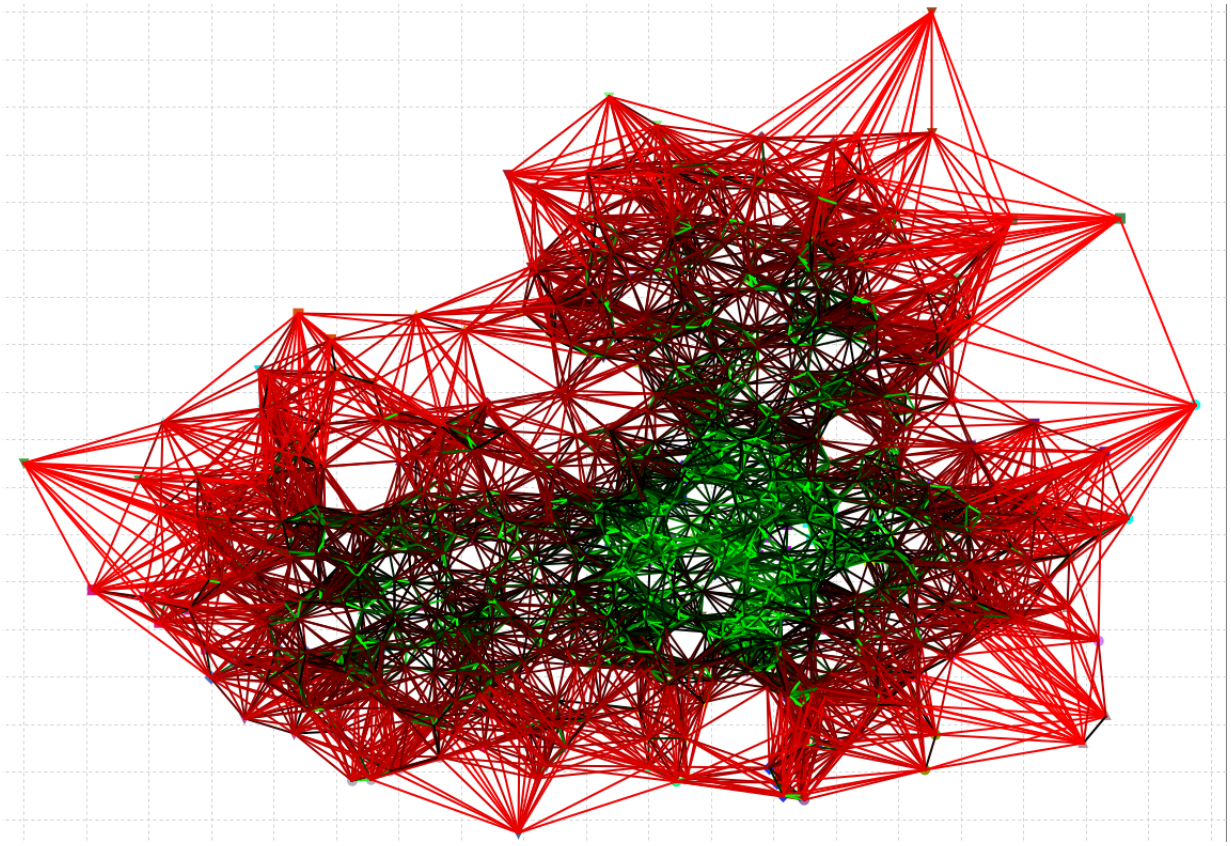


Figure A.6: Inspection tool implemented in Clueminer enables to explore individual steps of proposed Chameleon2 algorithm. In this particular case you can display a graph constructed from data vectors using their k-similarities.

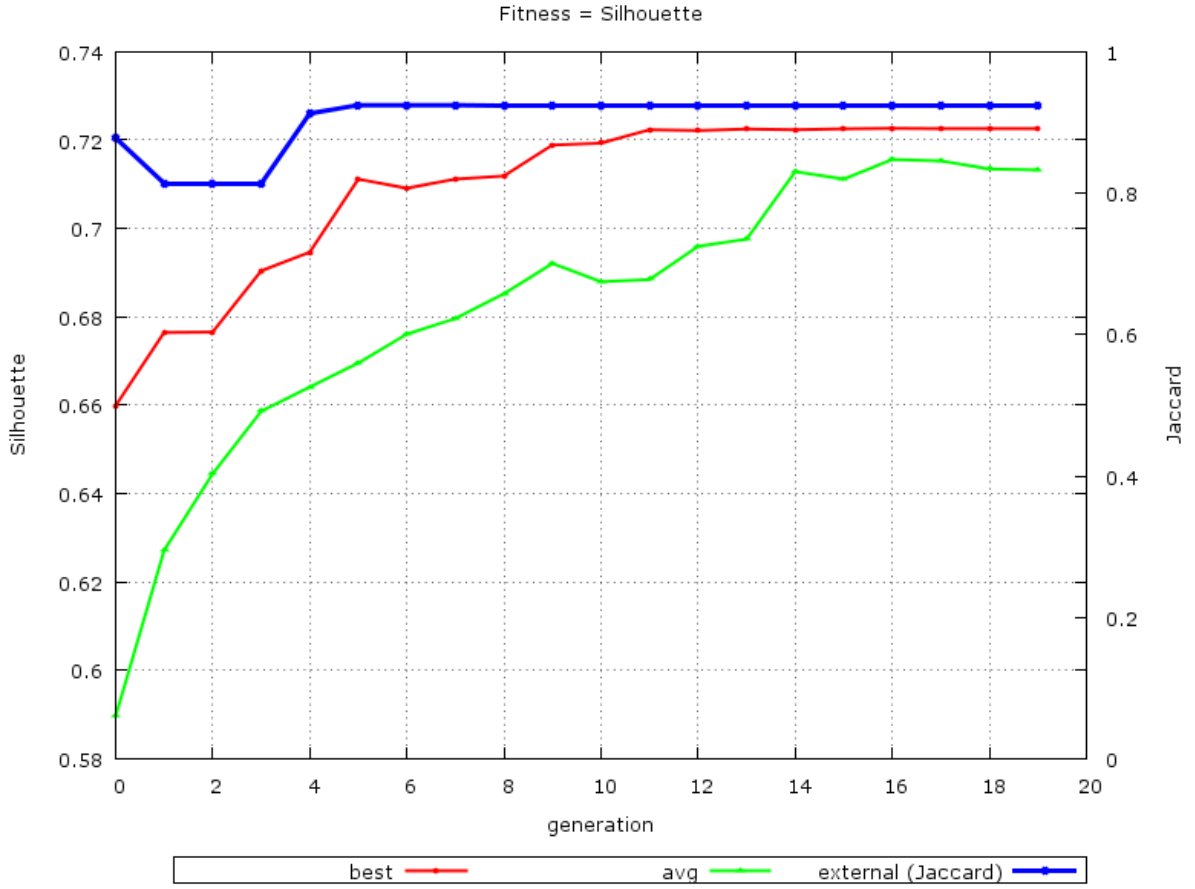


Figure A.7: Evolutionary algorithm explore the space of possible clusterings (and configurations of clustering algorithms) with an objective to maximize the Silhouette criterion (in this particular case). External criterion (overlap with external labels) is not improving much during the search.

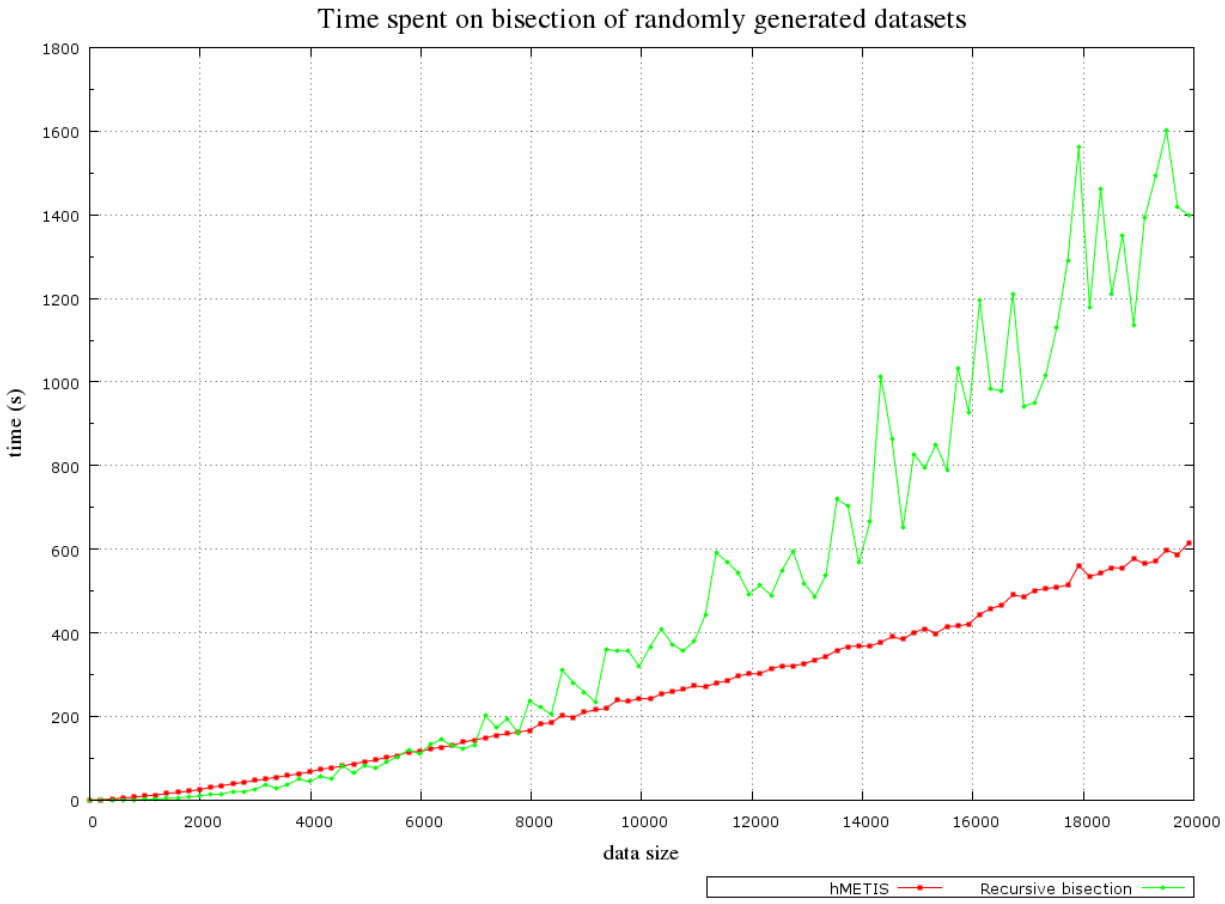


Figure A.8: We have made a lot of experiments enabling us to improve scalability of clustering algorithms implemented in Clueminer.