Clustering Based on Weighted Ensemble

A dissertation presented

by

Nejc Ilc

to

The Faculty of Computer and Information Science

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the subject of

Computer and Information Science

Ljubljana, 2016

Investing in your future
Operation Part Financed By The European Union
European Social Fund

Clustering Based on Weighted Ensemble

A dissertation presented
by

Nejc Ilc

to
The Faculty of Computer and Information Science
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy
in the subject of
Computer and Information Science

Ljubljana, 2016

APPROVAL

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

— Nejc Ilc —

October 2016

PREVIOUS PUBLICATION

I hereby declare that the research reported herein was previously published/submitted for publication in peer reviewed journals or publicly presented at the following occasions:

[1] N. Ilc, A. Dobnikar. Gravitational clustering of the self-organizing map. In A. Dobnikar, U. Lotrič, and B. Šter, editors, *Adaptive and Natural Computing Algorithms*, volume 6594 of *Lecture Notes in Computer Science*, pages 11–20, Berlin Heidelberg, 2011. Springer-Verlag. doi: 10.1007/978-3-642-20267-4_2

[2] N. Ilc, A. Dobnikar. Generation of a clustering ensemble based on a gravitational self-organising map. *Neurocomputing*, 96:47–56, 2012. doi: 10.1016/j.neucom.2011.10.043

[3] N. Ilc. Modified Dunn's cluster validity index based on graph theory. *Przegląd Elektrotechniczny (Electrical Review)*, 88(2):126–131, 2012. URL: http://pe.org.pl/articles/2012/2/36.pdf

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Ljubljana.

To Petra and Sofija

"The kingdom of heaven is like what happens when a net is thrown into a lake and catches all kinds of fish. When the net is full, it is dragged to the shore, and the fishermen sit down to separate the fish. They keep the good ones, but throw the bad ones away."

— Jesus Christ *(Matthew 13: 47–48)*

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Nejc Ilc
Razvrščanje z uporabo uteženega ansambla

POVZETEK

Razvrščanje podatkov v gruče je slabo pogojen problem in dokazano je, da algoritem, ki bi izpolnjeval vse predpostavke dobrega razvrščanja, ne obstaja. To je glavni razlog za obstoj velikega števila algoritmov za razvrščanje, ki temeljijo na raznovrstnih teoretičnih osnovah – med njimi je tudi znan algoritem Kohonenove samo-organizirajoče mreže (SOM). Na žalost naučena mreža SOM ne ponudi eksplicitno izražene strukture gruč v podatkih, zato navadno uporabimo dodaten korak, na katerem združujemo posamezne enote mreže v gruče. V disertaciji predstavljamo doprinos k dvonivojskemu razvrščanju z mrežo SOM, pri čemer uporabljamo principe zakona gravitacije. Predlagan algoritem za gravitacijsko razvrščanje samo-organizirajoče mreže (gSOM) je sposoben odkriti gruče poljubne in ne zgolj hipersferične oblike. Poleg tega algoritem gSOM sam določi število gruč v podatkih. Opravili smo primerjavo z nekaterimi drugimi tehnikami razvrščanja na umetnih in realnih podatkih. Izkaže se, da gSOM doseže obetavne rezultate, še posebej na podatkih o izraženosti genov.

Algoritem, ki bi znal rešiti vse probleme razvrščanja ne obstaja, zato je koristno analizirati podatke skozi večkratno razvrščanje. Pri tem nastane množica razvrstitev, ki tvorijo ansambel razvrstitev. Metode ansamblov za razvrščanje so se pojavile nedavno kot učinkovit pristop k stabilizaciji in izboljšanju delovanja enostavnih algoritmov za razvrščanje. Razvrščanje z ansambli je v osnovi sestavljeno iz dveh korakov: gradnja ansambla razvrstitev z enostavnimi metodami in združevanje dobljenih rešitev v sporazumno razvrstitev podatkov. Da bi olajšali korak združevanja v sporazum, je bil predlagan postopek uteževanja razvrstitev v ansamblu, ki skuša ovrednotiti pomembnost posameznih članov ansambla. Eden od načinov za analizo pomembnosti razvrstitev (PRA) je uporaba notranjih kazalcev veljavnosti razvrstitev. Na tem področju smo napravili dva prispevka: najprej predlagamo nov notranji ocenjevalni kazalec, imenovan DNs, ki razširja Dunnov kazalec in je osnovan na iskanju najkrajših poti v Gabrielovem grafu nad podatki; drugi prispevek je povezan z nadgradnjo obstoječega pristopa uteženega ansambla z dodatnim korakom redukcije, ki sledi koraku ocenjevanja razvrstitev v ansamblu. Razvit postopek analize pomembnosti razvrstitev z redukcijo (PRAr) se obnese zadovoljivo, ko ga vključimo v tri funkcije za iskanje sporazumne razvrstitve, pri čemer vse funkcije temeljijo na principu kopičenja dokazov.

V disertaciji se dotikamo vseh glavnih področij razvrščanja podatkov: ustvarjanje podatkov, analiza podatkov z enostavnimi algoritmi za razvrščanje, ocenjevanje razvrstitev z notranjimi in zunanjimi kazalci veljavnosti ter razvrščanje z ansambli s poudarkom na uteženih različicah. Vse predlagane doprinose smo primerjali s trenutno aktualnimi metodami na podatkih iz različnih problemskih domen. Rezultati kažejo na uporabnost predlaganih metod v kontekstu strojnega učenja.

Ključne besede: razvrščanje v gruče, nenadzorovano učenje, metode uteženega ansambla, ocenjevanje razvrstitev, generator umetnih podatkov

University *of Ljubljana*
Faculty *of Computer and Information Science*

Nejc Ilc
Clustering Based on Weighted Ensemble

ABSTRACT

The clustering is an ill-posed problem and it has been proven that there is no algorithm that would satisfy all the assumptions about good clustering. This is why numerous clustering algorithms exist, based on various theories and approaches, one of them being the well-known Kohonen's self-organizing map (SOM). Unfortunately, after training the SOM there is no explicitly obtained information about clusters in the underlying data, so another technique for grouping SOM units has to be applied afterwards. In the thesis, a contribution towards a two-level clustering of the SOM is presented, employing principles of Gravitational Law. The proposed algorithm for gravitational clustering of the SOM (gSOM) is capable of discovering complex cluster shapes, not only limited to the spherical ones, and is able to automatically determine the number of clusters. Experimental comparison with other clustering techniques is conducted on synthetic and real-world data. We show that gSOM achieves promising results especially on gene-expression data.

As there is no clustering algorithm that can solve all the problems, it turns out as very beneficial to analyse the data using multiple partitions of them – an ensemble of partitions. Cluster-ensemble methods have emerged recently as an effective approach to stabilize and boost the performance of the single-clustering algorithms. Basically, data clustering with an ensemble involves two steps: generation of the ensemble with single-clustering methods and the combination of the obtained solutions to produce a final consensus partition of the data. To alleviate the consensus step the weighted cluster ensemble was proposed that tries to assess the relevance of ensemble members. One way to achieve this is to employ internal cluster validity indices to perform partition relevance analysis (PRA). Our contribution here is two-fold: first, we propose a novel cluster validity index DNs that extends the Dunn's index and is based on the shortest paths between the data points considering the Gabriel graph on the data; second, we propose an enhancement to the weighted cluster ensemble approach by introducing the reduction step after the assessment of the ensemble partitions is done. The developed partition relevance analysis with the reduction step (PRAr) yields promising results when plugged in the three consensus functions, based on the evidence accumulation principle.

In the thesis we address all the major stages of data clustering: data generation, data analysis using single-clustering algorithms, cluster validity using internal and external indices, and finally the cluster ensemble approach with the focus on the weighted variants. All the contributions are compared to the state-of-art methods using datasets from various problem domains. Results are positive and encourage the inclusion of the proposed algorithms in the machine-learning practitioner's toolbox.

Key words: cluster analysis, unsupervised learning, weighted cluster ensemble, cluster validation, synthetic data generation

ACKNOWLEDGEMENTS

I am very grateful for all people who helped me during this long journey. First, I would like to thank my advisor, professor Andrej Dobnikar for his persistent support, guidance and stimulation. Our debates over academic and personal matters widened my perception of the world. I would also like to thank other members of the doctoral committee for doing their hard job reviewing the thesis and giving me constructive suggestions to improve it.

Many thanks to members of the Laboratory of Adaptive Systems and Parallel Processing; I am very pleased that we spend these years together as friends. Thank you for all the jokes and laughter that make our job and life nicer. Special thanks to Erik Štrumbelj for his generous help on analysing the results using Bayesian statistic modelling. The cover image was created by young artists Ana Janež and Črt Mate, thank you both for sharing your talent with me. Sister Helena, thank you for doing a grammar check of the Slovenian part.

This thesis would not exist without the support of my beloved wife Petra and my entire family. You raised me up when I was hopeless and without any strengths to go further. Thank you for believing in me. Dear daughter Sofija, I was finishing the draft when you came into this world and showed me what really matters in life – thank you.

Finally, thank you my Lord, Teacher, and Brother – for everyone and everything.

— Nejc Ilc, Ljubljana, October 2016.

CONTENTS

Introduction

This book may save your life. – Most probably not by reading it or throwing its hard copy to the attacker in a self-defence. Well, you can try, indeed. Our point is somewhere else – in this thesis we address fundamental algorithms for data analysis that are used in nearly any field of science you can imagine. This means, they are also implemented in the real-life and control the machines, devices, and services you probably use every day. *These algorithms* are processing tons of data every moment, even when you are sleeping. *These algorithms* enable experts to see through the unstructured data and to find the needle in a haystack. Who knows, maybe a newly discovered group of genes with similar function would lead us to better understand the diseases and how to prevent them? Can *these algorithms* help us with that? Certainly. The algorithms we are talking about are the data clustering algorithms.

1.1 Motivation and background

Nowadays, our life is crammed with data of all kinds. We want to interpret these data in order to gain some knowledge. Data clustering is used as a fundamental tool in the fields of machine learning, pattern recognition [1], and data mining [2, 3], where an efficient analysis, visualization and interpretation of the data is very important, especially due to the constant growth in the volume of data. Cluster analysis has a long and rich history that spans over more than 60 years now [4] and its development has not declined since. Understandable enough, for the need for automatic data processing increases with each innovation in information communication technology. Nowadays, a machine-learning practitioner's toolbox contains dozens of various algorithms for data analysis and clustering algorithms are usually the first he or she applies on the data to get familiar with them. Numerous applications of clustering are made, ranging from image segmentation, text mining, gene expression analysis, study of social phenomena, and market analysis to remote sensing in geology, taxonomy analysis in biology, pathology research, and diagnosis in medicine [5].

Clustering is a process of organizing data into natural groups or clusters, such that similar data points are assigned to the same cluster [6]. Data clustering is known as an unsupervised learning task, meaning that in contrast to supervised learning, the number of clusters is not predefined and none of the input data points is labelled [7]. Despite the feeling that clustering seems to be conceptually very close to us, it has been proven by Kleinberg that there is no clustering algorithm that would satisfy all the assumptions about good clustering [8]. This is why the body of the literature constantly grows by the new approaches and modifications of already established clustering algorithms.

On the one hand, clustering is a very intuitive and universal task, but on the other hand, aforementioned Kleinberg's impossibility theorem implies there is no clustering algorithm that can solve all the problems. Consequently, it turns out as very beneficial to consider analysis of data with a set of different clustering methods. We say that two heads are better than one and we more likely trust a committee of experts than an individual. This is also the core notion of the ensemble approach [9]. Basically, the reasoning behind any type of ensemble is to look on a problem from different perspectives in order to maximize the performance of some kind. Hence, one has to solve the problem of merging multiple partitions or clusterings into one common result, called the consensus partition. The methods that do this are called consensus functions. In short, this is what the cluster ensembles are about [10].

Let us imagine we have clustered some data few times and we would like to get the consensus out of the partitions. The question arises, whether the all partitions are of the same *goodness*. If not, why should they have all the same importance? Here comes in the play the sub-field of cluster analysis called the cluster validation [11]. A cluster validity index (CVI) is a measure of *goodness* for a certain partition [12]. Basically, we categorize cluster validity indices into the internal and external ones. The latter compute the agreement between the partition in question and the ground-truth or reference partition known in advance. Therefore, they are suitable as the objective performance measures. The internal cluster validity index is not aware of any reference partition or external information except for the dataset and the output of clustering algorithm. We use the internal validation when there is no ground-truth and this is most often the case. So, before we start computing the consensus partition, maybe it would be a good idea to evaluate each partition in the ensemble and measure its goodness with the internal validity measure. Then, we could use this information to appropriately weight the partitions in the process of their consolidation. This is the summary of the Partition Relevance Analysis (PRA), first defined by Duarte et al. [13], although they did not use this name. They proposed the approach, where multiple CVIs are involved in the process of evaluation and thereafter their averages are considered as weights of partitions in the ensemble [14]. An approach where partitions in the ensemble are weighted is called the *weighted cluster ensemble*.

If we want to analyse our data to find clusters in them using the weighted cluster ensemble, we are to consider at least these three steps:

1. Generate the ensemble of partitions from data using a single-clustering algorithm (or more).

2. Weight those partitions using an internal cluster validity index (or more).

3. Apply a consensus function on the ensemble considering the weights of partitions.

In our work, we address all three steps and try to "leave the world better than we found it" (Robert Baden-Powell).

1.2 Contributions to science

The main scope of the thesis is oriented towards the improvements of existing approaches to data clustering on the level of data analysis with a single-clustering algorithm, internal validation of data clustering, and the consolidation of data clusterings using the ensemble approach. We summarize our contributions to the scientific community in the following.

- *Development of a novel clustering method based on a self-organizing neural network and gravitational clustering.* We propose the gSOM algorithm that tackles the data clustering problem using a two-level approach [15]. First, data are abstracted using the self-organizing map (SOM) algorithm proposed by Kohonen [16]. The SOM neurons are then clustered into final solution using a nature-inspired algorithm based on the gravitational principle [17]. The gSOM algorithm estimates the number of clusters automatically. The evaluation on synthetic, genetic, and other real-world datasets is conducted proving the effectiveness of the proposed algorithm. Moreover, gSOM is used as an ensemble generator [18].

- *Proposal of a novel cluster validity index, based on a graph theory, for weighting the partitions in the ensemble.* We devised a modification to the well-known Dunn's internal validity index [19]. Our proposal, the DNs index, is based on the shortest paths between the data points considering the Gabriel graph on the data [20]. From the literature we selected 33 other indices and compared them with the DNs index on the synthetic as well as on real-world datasets of various types. To the best of our knowledge it is the first study that systematically addresses the performance of CVIs in the case of datasets with linearly non-separable clusters. All the indices are then employed in the partition relevance analysis of the weighted cluster ensemble approach.

- *Enhancement of the weighted cluster ensemble approach using the partition relevance analysis with the reduction step.* How to select the cluster validity indices and how many of them to select are two main questions when we design the partition relevance analysis scheme (PRA). We address those questions by proposing the enhancement to the PRA using the reduction step (PRAr), where the CVIs are selected using the feature selection and extraction methods. The PRAr effectiveness is evaluated using the modification of three well-established consensus functions. A comprehensive experimental study is conducted including a multitude of different configurations for ensemble-generation step and for the PRAr itself.

- *Novel synthetic data generator with a control over the linear-separability.* To compare and evaluate different stages of cluster analysis, we developed the algorithm for two-dimensional data generation. It is capable of creating clusters of complex shapes with controllable minimal distance between clusters and the degree of linear-separability. A family of datasets called `Complex2D` is generated and used as a benchmark for the comparison of single-clustering algorithms, cluster validity indices, and cluster ensemble approaches.

For the purposes of this thesis we developed the MATLAB toolbox for cluster analysis called Pepelka[1]. It can be downloaded from `http://laspp.fri.uni-lj.si/nejci/Pepelka`.

1.3 Thesis outline

We begin the thesis with the essential step in cluster analysis, namely the data generation – without the data there is no data clustering. In Chapter 2 we introduce the background of data synthesis and present a novel algorithm for the construction of

[1]*Pepelka* is a Slovenian word for Cinderella, a heroine of a well-known fairy tale.

clusters of complex shapes. Thus, we named the resulting two-dimensional dataset family as `Complex2D`. We proceed with the cluster validation techniques in Chapter 3. There we review 33 internal cluster validity indices and propose a Dunn-like index based on the shortest paths in the graph, DNs in short. We compare DNs with other indices using four complementary evaluation methodologies. We address the algorithms for data clustering in Chapter 4, where we focus on the two-level methods based on self-organizing map. We propose the clustering of the SOM using the simulation of the gravitational field, which results in the gSOM algorithm. We compare the gSOM with the seven other algorithms using a statistical test suitable for the pairwise comparisons over multiple datasets. We discuss the cluster ensemble framework with partition relevance analysis in Chapter 5. We introduce the reduction step in the PRA and experimentally show its advantages. We round up our story of data clustering in Chapter 6 with further research directions in mind.

1.4 Notation

1.4.1 List of symbols and notation

Here, we list the symbols and notation we use throughout the thesis. Vectors, matrices, and sets are put in bold.

$\mathbf{X}$	set of data points, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$
$\mathbf{x}$	D-dimensional data point; $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} \subset \mathbb{R}^D$
D	number of dimensions or features of data point
N	number of data points, $N = \lvert\mathbf{X}\rvert$
$\overline{\mathbf{X}}$	centroid of data set $\mathbf{X}$, i.e. its mean vector, $\overline{\mathbf{X}} = \frac{1}{N}\sum_{\mathbf{x}_i \in \mathbf{X}} \mathbf{x}_i$
L	number of cluster pairs that are linearly non-separable
d_{min}	minimal distance between every cluster pair
U	number of neuron units in the self-organizing map
S	scale factor that determines the number of neurons U
σ	width of a Gaussian kernel

G	gravitational constant
ΔG	proportion of reducing the value of G
$\mathbf{d}$	distance vector between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$
$d_E(\mathbf{x}_i, \mathbf{x}_j)$	Euclidean distance between data points $\mathbf{x}_i$ and $\mathbf{x}_j$, also denoted as $\|\|\mathbf{d}\|\|$
$\mathbf{Q}$	set of remaining particles at current iteration of gSOM
$\mathbf{G}$	a graph $\mathbf{G}$ with a set of vertices $\mathbf{V}$ and a set of edges $\mathbf{E}$, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
$\mathbf{c}$	cluster of data points
n_i	number of data points in the cluster $\mathbf{c}_i$, i.e. $n_i = \|\mathbf{c}_i\|$
$\overline{\mathbf{c}_i}$	centroid of a cluster $\mathbf{c}_i$, i.e. its mean vector, $\overline{\mathbf{c}_i} = \frac{1}{n_i} \sum_{\mathbf{x}_i \in \mathbf{c}_i} \mathbf{x}_i$
n_{hk}	number of data points that are common for clusters $\mathbf{c}_h$ and $\mathbf{c}_k$
$\mathbf{N}$	a set of clusters sizes, i.e. $\mathbf{N} = \{n_1, n_2, \dots, n_K\}$
$\mathbf{C}$	partition or clustering of dataset $\mathbf{X}$ with K clusters; $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K\}$
$\mathbf{C}^T$	true partition of dataset provided as ground-truth
$\mathbf{C}^*$	the optimal partition selected by a CVI with K^* clusters
$\hat{\mathbf{C}}$	the optimal partition selected by an eCVI
K	number of clusters, $K = \|\mathbf{C}\|$
K_T	true number of clusters provided as ground-truth, $K_T = \|\mathbf{C}^T\|$
$\mathbf{P}$	cluster ensemble with M members, $\mathbf{P} = \{\mathbf{C}_1, \mathbf{C}_2, \dots \mathbf{C}_M\}$
M	number of partitions in a cluster ensemble, $M = \|\mathbf{P}\|$
$\mathbf{C}^\mathbf{P}$	consensus partition obtained by consensus function on $\mathbf{P}$
$\mathbf{S}$	similarity matrix between data points based on a cluster co-occurence
$\mathbf{w}$	vector of weights
$\mathbf{r}$	matrix of raw values of internal cluster validity indices
$\mathbf{R}$	matrix of unified values of internal cluster validity indices
$\hat{\mathbf{R}}$	reduced matrix $\mathbf{R}$
Γ	unification function
Π	reduction function

Ω	aggregation function

1.4.2 *List of acronyms*

We list the acronyms defined in the thesis in alphabetical order.

AL	Average-Linkage hierarchical clustering algorithm
AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
ART	Adaptive Resonance Theory neural network
BCA	Balanced Clustering Accuracy
CBK	Clustering with Background Knowledge
CL	Complete-Linkage hierarchical clustering algorithm
CLK	Clustering with Lack of Knowledge
CSPA	Cluster-based Similarity Partitioning Algorithm
CSPA-W	Weighted Cluster-based Similarity Partitioning Algorithm using PRAr
CVI	internal Cluster Validity Index
DANCo	Dimensionality from Angle and Norm Concentration method for data dimensionality estimation
DICLENS	DIvisive Clustering ENSemble with automatic cluster number
DICLENS-W	Weighted DIvisive Clustering ENSemble with automatic cluster number using PRAr
EAC	Evidence Accumulation Clustering
EAC-W	Weighted Evidence Accumulation Clustering using PRAr
eCVI	external Cluster Validity Index
ESOM	Emergent Self-Organizing Map
FEKM	Feature Extraction using K-Means algorithm
FSKM	Feature Selection using K-Medoids algorithm
GSA	Gravitational Search Algorithm
gSOM	gravitational clustering of Self-Organizing Map
HGPA	HyperGraph Partitioning Algorithm
HSOM	Hierarchical Self-Organizing Map

JWEAC	Joint Weighted Evidence Accumulation Clustering
KM	*K*-Means clustering algorithm
kNN	*k*-nearest neighbour algorithm
LCE	Link-based Cluster Ensemble
LS	Laplacian Score feature selection method
MCLA	Meta-CLustering Algorithm
PAC	Probability Accumulation Clustering
PMML	Predictive Model Mark-up Language
PPCA	Probabilistic Principal Component Analysis
PRA	Partition Relevance Analysis
PRAr	Partition Relevance Analysis with reduction step
PSDG	Parallel Synthetic Data Generator
RGC	Randomized Gravitational Clustering
RRA	Robust Rank Aggregation
SDDL	Synthetic Data Definition Language
SL	Single-Linkage hierarchical clustering algorithm
SL-SOM	Self-Labelling Self-Organizing Map
SMST	Similarity-based Minimum-cost Spanning Tree
SOM	Self-Organizing Map
SOM-CCC-MDC	Contiguity-Constrained Clustering of Self-Organizing Map using Minimal-Distance Criterion
SOM-CCC-MVC	Contiguity-Constrained Clustering of Self-Organizing Map using Minimal-Variance Criterion
SOMKm	clustering of Self-Organizing Map with *k*-means
SOMNcut	clustering of Self-Organizing Map with Normalized cuts
SOMSpec	Spectral clustering of Self-Organizing Map
SOMStar	clustering of Self-Organizing Map using connected components
SWEAC	Single Weighted Evidence Accumulation Clustering
Sp	Spectral clustering algorithm
Spec	Spectral feature selection method
WEA	Weighted Evidence Accumulation
WEAC	Weighted Evidence Accumulation Clustering

Synthetic data generation

2.1 Introduction

The ultimate goal of machine learning and computer science in general is to solve real-world problems, measurable in some way. So, we develop algorithms that are applied on the "real" data – these are basically recorded values of the particular features we identify when tackling a problem. However, in a process of data analysis we are sometimes able to discover the characteristics and structure of data, but we cannot manipulate with them. This is the reason some researchers create their own data in a controllable manner. We call this procedure a synthetic or artificial data generation. It is essentially useful when benchmarking specific features of learning algorithms.

One of the main challenges in cluster analysis, as stated by the authors of the Fundamental Clustering Problems Suite [21], are clusters that cannot be separable by a hyperplane – we say they are linearly non-separable. In other words, cluster of data points lies partially or completely inside the convex hull of other cluster. To the best of our knowledge, no data generator has been devised so far that is able to control the amount of linear separability between clusters. Our contribution, a data generator, fills this void and opens the possibility to systematically study and compare the performance of cluster analysis algorithms on arbitrary-shaped clusters with controllable distances and amount of linear separability between them.

It is surprising how little of published research in the field of data mining had addressed systematic and reproducible synthetic data generation before the work of Pei and Zaïane [22] in 2006, who developed a versatile data generator for the purpose of assessing the algorithms for clustering and outlier detection. Their generator outputs two-dimensional numeric data and enables a user to control the number and density of data points in each cluster, the number of clusters in dataset, the cluster shapes classified into five difficulty levels, the density function, and the level of background noise. The authors also published a Java implementation with graphical user interface[1]. The generator allows us to set the distance between cluster means indirectly by adjusting difficulty and density levels of clusters. Consequently, the clusters in the generated dataset often overlap, as we cannot define a minimal gap between the nearest data points in the adjacent clusters. We illustrate this issue in Fig. 2.1, showing a screenshot of Pei and Zaïane data generator user interface, where five overlapping clusters were generated.

[1]Available at http://webdocs.cs.ualberta.ca/~yaling/Cluster/Php/data_gen.php.

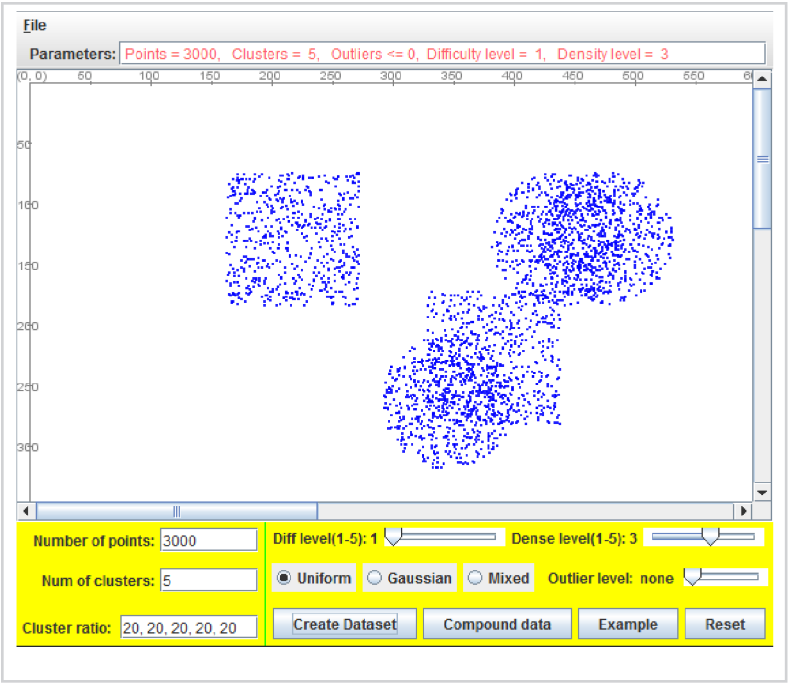


Figure 2.1

Problem of cluster overlap when using Pei and Zaïane data generator.

Hoag and Thompson devised Parallel Synthetic Data Generator (PSDG) that is intended for large datasets of terabytes size and utilizes the power of modern multiple-processor systems [23]. Produced datasets are described in the XML-based format called Synthetic Data Definition Language (SDDL). Recently, Alexandrov et al. proposed an improvement over PSDG, called Myriad [24].

When the purpose of a data generator is to mimic or reproduce real-world problems, one may find it beneficial to exploit a model of real data, which is what Eno and Thompson did [25]. They proposed to build a model of input data with decision trees and describe both the data and the model with Predictive Model Mark-up Language (PMML). Their main contribution is a mapping between PMML and SDDL, which is able to generate synthetic data of any size with similar patterns as in the original data. Similar concept is investigated by Marko Robnik-Šikonja [26] who proposed a semiartificial data generator based on radial basis function neural networks.

Adä and Berthold developed the Modular Data Generator – an extension module of a visual data-mining tool KNIME [27]. This module can be easily incorporated into a data-processing work-flow and can be easily extended with new processing modules. It is quite universal and covers data generation for various problem domains, like shopping-basket analysis, association rules, and cluster analysis. We are especially interested in the latter; with Modular Data Generator one can generate hyper-ellipsoidal Gaussian clusters as well as clusters of complex shapes for it supports arbitrary user-defined formula. The shortcoming of this generator is that we cannot control the distance between the clusters dynamically as the positions of clusters have to be predefined.

Frasch et al. proposed a data generator for classification with controllable statistical properties [28] – the emphasis is on the Bayes error rate [29]. Data are generated by white Gaussian densities with their means on the corner of a regular k-simplex, e.g. vertices of a triangle in a 2-dimensional space. We found it rather inappropriate for clustering tasks due to the generation of the overlapping clusters. Also, the generated clusters are at most hyper-ellipsoidal and therefore of a moderate difficulty level.

2.2 *Data generator with a control over linear separability*

We developed a synthetic data generator that is able to produce two-dimensional data sets with versatile cluster[2] shapes. A user can specify the number of clusters, the number of data points in each cluster, the shape of clusters, the distribution of data points within cluster shape, the minimal distance between clusters, and the degree of linearly separable clusters, i.e. how many cluster pairs are separable by a line or a hyperplane in general. We focus ourselves to the modelling of datasets with different levels of interaction between clusters. This is one of the issues in cluster analysis but certainly not the only one – for example, noise and outliers represent another difficulty level, but we will not address it with our generator for now. Nevertheless, we assume that noise and outliers are included in the real-world datasets we use in experiments.

To test whether two clusters are linearly separable, we employ linear programming using the Simplex method[3] [30]. We are searching for a hyperplane H in $\mathbb{R}^D$ such

[2]As we are discussing unsupervised learning, a "cluster" may be a more suitable word than a "class". However, those terms are exchangeable if we do not consider the application of generated data.

[3]Our explanation here is partly following an excellent post published by Raffael Vogler at http://www.joyofdata.de/blog/testing-linear-separability-linear-programming-r-glpk.

that it separates data points in one cluster from another. If we find such a hyperplane, these clusters are linearly separable and if not, we say they are linearly non-separable.

Let's have clusters $\mathbf{c}_U = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{N_U}\}$ and $\mathbf{c}_V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{N_V}\}$, where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and $N = N_U + N_V$. We define a hyperplane as $H = \{\mathbf{x} : \mathbf{h}^T\mathbf{x} = \beta\}$, where $\mathbf{h}, \mathbf{x} \in \mathbb{R}^D$ and $\beta \in \mathbb{R}$.

If the clusters $\mathbf{c}_U$ and $\mathbf{c}_V$ are linearly separable, a β and $\mathbf{h}$ exist such that the following two inequalities hold:

$$\forall \mathbf{u} \in \mathbf{c}_U \quad : \quad \mathbf{h}^T\mathbf{u} > \beta\,, \tag{2.1}$$

$$\forall \mathbf{v} \in \mathbf{c}_V \quad : \quad \mathbf{h}^T\mathbf{v} < \beta\,. \tag{2.2}$$

We transform strict inequalities to non-strict using an arbitrary small ϵ. Then we multiply both inequalities with $1/\epsilon$ and move all the variables on the left hand side. Finally, we multiply the Eq. (2.1) with -1 to unify the direction of the inequalities and to comply with the standard form of inputs for the MATLAB linear programming solver:

$$\mathbf{h}^T\mathbf{u} \geq \beta + \epsilon \quad \Rightarrow \quad -\frac{\mathbf{h}^T}{\epsilon}\mathbf{u} + \frac{\beta}{\epsilon} \leq -1\,, \tag{2.3}$$

$$\mathbf{h}^T\mathbf{v} \leq \beta - \epsilon \quad \Rightarrow \quad \frac{\mathbf{h}^T}{\epsilon}\mathbf{v} - \frac{\beta}{\epsilon} \leq -1\,. \tag{2.4}$$

We can substitute $\beta' = \beta/\epsilon$ and $\mathbf{h}' = \mathbf{h}/\epsilon$. So, we are trying to find a hyperplane $H' = \{\mathbf{x} : \mathbf{h}'^T\mathbf{x} = \beta'\}$and if one exists, also the hyperplane H exists. We get the final form of linear programming problem:

$$-\mathbf{h}'^T\mathbf{u} + \beta' \quad \leq \quad -1\,, \tag{2.5}$$

$$\mathbf{h}'^T\mathbf{v} - \beta' \quad \leq \quad -1\,. \tag{2.6}$$

Now, we employ the Simplex method for solving the linear programming problem, which can be formulated as an optimization problem:

$$\min_{\mathbf{x}} \mathbf{f}^T\mathbf{x}, \text{ such that } \underset{N\times(D+1)}{\mathbf{A}} \cdot \underset{(D+1)\times 1}{\mathbf{x}} \leq \underset{N\times 1}{\mathbf{b}}\,, \tag{2.7}$$

where

$$\mathbf{A} = \begin{bmatrix} -u_1^{(1)} & -u_1^{(2)} & \cdots & -u_1^{(D)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -u_{N_U}^{(1)} & -u_{N_U}^{(2)} & \cdots & -u_{N_U}^{(D)} & 1 \\ -v_1^{(1)} & -v_1^{(2)} & \cdots & -v_1^{(D)} & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -v_{N_V}^{(1)} & -v_{N_V}^{(2)} & \cdots & -v_{N_V}^{(D)} & -1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} h'^{(1)} \\ h'^{(2)} \\ \vdots \\ h'^{(D)} \\ \beta' \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}, \tag{2.8}$$

and the linear objective function vector **f** set to all zeros, which means we do not want to optimize the distance between the cluster members and the plane H', but only want to find one if exists:

$$\mathbf{f} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{D+1}. \tag{2.9}$$

2.2.1 Cluster shapes

We have to define the "body" of cluster – its borders in order to manipulate distance between clusters and linear separability. Here we employ α-shapes [31, 32] as a tool for describing the shape or silhouette of a set of points in the plane. The α-shape is a generalization of the convex hull parametrized by the real non-negative number α that controls the level of details of the shape. The smaller α is, the more detailed the shape of a cluster gets, and vice-versa. So, when $\alpha \to \infty$ the α-shape becomes the convex hull of the data points in the cluster. We define an α-shape as follows: if the bounding circle of an empty open disk with radius α passes through two points then there is an edge connecting these two points; α-shape is a set of all the edges that satisfies this condition. We set the value for α as the smallest disk radius that produces an α-shape with only one region. Furthermore, to avoid fragmented shapes with small holes in the middle we define an area threshold under which the holes are suppressed. In our experiments, the 1% of a bounding-box area suffice.

Our generator can produce 38 different cluster shapes, some of them are depicted in Fig. 2.2. The majority of the shapes are identical to those defined by Pei and Zaïane [22] and demonstrate different levels of complexity: compact and spherical; elongated; with corners and holes that enable embedding of other clusters. Cluster members, i.e.

data points, can be drawn from the uniform or truncated normal distribution; the user can choose from the two or allow the algorithm to pick one at random, i.e. mixed distribution.

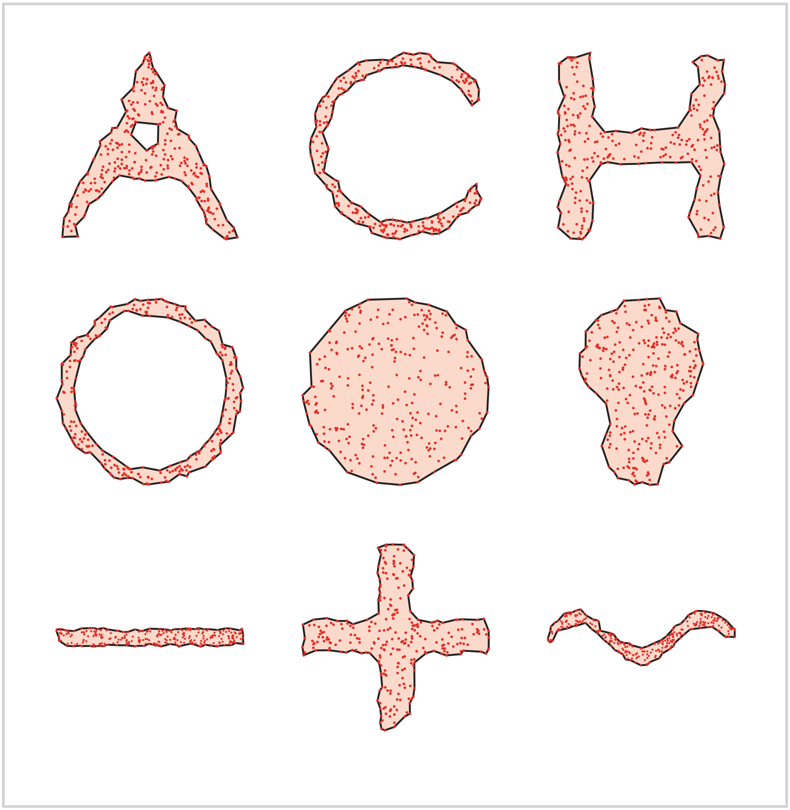


Figure 2.2

Some examples of cluster shapes. Red dots represent data points, bright red patch represents the body of the cluster that is defined by α-shape boundary drawn as black outline. Each cluster contains 300 data points generated from the uniform distribution.

2.2.2 Data generation algorithm

The main feature of our proposal is controlling the distance and linear separability between clusters. Besides, our generator produces two-dimensional data with non-overlapping, crisp clusters. This enables us simple and efficient visual inspection. It also features a "batch mode", which means it can automatically generate a collection of datasets by varying its parameters.

We developed an iterative algorithm that tries to achieve this in a two-step procedure: first, a new cluster is created and is being moved towards the existing ones until it collides with one of them; second, fine-tuning of the new cluster's position is done by random rotations and translations. The proposed algorithm and its inputs are discussed in greater details in the following.

Input parameters:

- K: number of clusters to generate,
- $\mathbf{N}$: number of data points in each cluster; a vector of K positive integers $\mathbf{N} = \{n_1, n_2, \dots, n_K\}$,
- d_{min}: desired minimal distance between every cluster pair,
- L: desired number of cluster pairs that are linearly non-separable.

Additional algorithm options:

- S: *stiffness* - how many percent of distance between selected cluster pair is reduced on each iteration on a coarse level [default: 0.5],
- I_{coarse}: the number of iterations in the first step, i.e. coarse level of movement [default: 300],
- I_{fine}: the number of fine-tuning iterations in the second step [default: 200],
- β_{coarse}: maximum angle of a random rotation in the first step [default: π],
- β_{fine}: maximum angle of a random rotation in the second step [default: $\frac{\pi}{2}$],
- `tol`: tolerance of d_{min} [default: 10% of d_{min}],
- `shapes`: list of cluster shapes,
- `distribution`: uniform or truncated normal distribution of data points in the clusters [default: uniform],
- r_{gen}: a distance from the centroid of the first-created cluster to the position of the newly generated cluster [default: 20].

A pseudocode of the data generator is listed in Algorithm 1, helper functions are defined in Algorithm 2.

Algorithm 1

Data generator with control over linear separability of clusters

Input: $\mathbf{N}$, K, $d_{\min}$, L
Output: $\mathbf{X}$

```
1:  $\mathbf{c}_1 \leftarrow$ ClusterCreate$(n_1)$
2:  $\mathbf{c}_1 \leftarrow$ ClusterRotate$(\mathbf{c}_1, \overline{\mathbf{c}_1}, 2\pi)$
3:  $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{c}_1$
4:  for $i \leftarrow 2$ to $K$
5:      $\mathbf{c}_i \leftarrow$ ClusterCreate$(n_i)$                                   ▷ $n_i \in \mathbf{N}$
6:      $\mathbf{c}_i \leftarrow$ ClusterRotate$(\mathbf{c}_i, \overline{\mathbf{c}_i}, 2\pi)$
7:      $\mathbf{b} \leftarrow$ random point on a boundary of a circle with radius $r_{\text{gen}}$
8:      $\mathbf{c}_i \leftarrow$ ClusterMove$(\mathbf{c}_i, \overline{\mathbf{c}_i}, \mathbf{b})$
9:      $\mathbf{c}_i^* \leftarrow \mathbf{c}_i$                                   ▷ Save this cluster position.
10:     for iter$_c \leftarrow 1$ to $I_{\text{coarse}}$                ▷ First step: coarse movement of a cluster.
11:         $\mathbf{c}_{\text{fix}} \leftarrow$ random cluster among 1 to $(i-1)$ clusters
12:         $\mathbf{m} \leftarrow \arg\min_{\mathbf{x}}\{\min_{\mathbf{y}} d_E(\mathbf{x},\mathbf{y})\}$, where $\mathbf{x} \in$ Bnd$(\mathbf{c}_i)$ and $\mathbf{y} \in$ Bnd$(\mathbf{c}_{\text{fix}})$
13:         $\mathbf{m}' \leftarrow \mathbf{m} + S \cdot (\overline{\mathbf{c}}_{\text{fix}} - \mathbf{m})$                ▷ $S$ is the *stiffness* parameter.
14:         $\mathbf{c}_i \leftarrow$ ClusterMove$(\mathbf{c}_i, \mathbf{m}, \mathbf{m}')$
15:         $\beta \leftarrow$ rnd$(-1,1) \cdot \beta_{\text{coarse}} \cdot d_E(\mathbf{m}', \overline{\mathbf{c}}_{\text{fix}}) \,/\, r_{\text{gen}}$
16:                         ▷ rnd$(-1,1)$ is a random real number on interval $(-1,1)$
17:         $\mathbf{c}_i \leftarrow$ ClusterRotate$(\mathbf{c}_i, \mathbf{m}', \beta)$
18:         isFineTuned $\leftarrow 0$
19:         for iter$_f \leftarrow 1$ to $I_{\text{fine}}$                       ▷ Second step: fine-tuning.
20:             $d \leftarrow \min_{\mathbf{x},\mathbf{y}}\{d_E(\mathbf{x},\mathbf{y})\}$, where $\mathbf{x} \in$Bnd$(\mathbf{c}_i)$ and $\mathbf{y} \in \bigcup_{k=1}^{i-1}$Bnd$(\mathbf{c}_k)$
21:             if $d < d_{\min}$                     ▷ If too close, move cluster $\mathbf{c}_i$ away.
22:                 $\mathbf{m}, \mathbf{f} \leftarrow \arg\min_{\mathbf{x},\mathbf{y}}\{d_E(\mathbf{x},\mathbf{y})\}$,
                        where $\mathbf{x}, \mathbf{m} \in$Bnd$(\mathbf{c}_i)$ and $\mathbf{y}, \mathbf{f} \in \bigcup_{k=1}^{i-1}$Bnd$(\mathbf{c}_k)$
23:                 $\mathbf{m}' \leftarrow \mathbf{m} + (\mathbf{m} - \mathbf{f}) \,/\, d \cdot |d - d_{\min}|$
24:                 $\mathbf{c}_i \leftarrow$ ClusterMove$(\mathbf{c}_i, \mathbf{m}, \mathbf{m}')$
25:                 $\beta \leftarrow$ rnd$(-1,1) \cdot \beta_{\text{fine}} \cdot (1 - \text{iter}_f \,/\, I_{\text{fine}})$
26:                 $\mathbf{c}_i \leftarrow$ ClusterRotate$(\mathbf{c}_i, \overline{\mathbf{c}}_i, \beta)$
```

```
27:             else
28:                 isFineTuned ← 1
29:                 break the loop                        ▷ End of fine-tuning.
30:             end if
31:         end for
32:         $l$ ← number of all linearly non-separable cluster pairs
33:         $l_i$ ← number of clusters that are linearly non-separable with $\mathbf{c}_i$
34:         if $\mathbf{c}_i$ overlaps with any other cluster OR isFineTuned == 0 OR $l > L$
35:             $\mathbf{c}_i \leftarrow \mathbf{c}_i^*$                   ▷ Reset cluster's move.
36:         end if
37:         stopDist ← 0                          ▷ Evaluate stopping criteria.
38:         stopLinNonSep ← 0
39:         if isFineTuned == 1 AND $|d - d_{\min}| \leq$ tol
40:             stopDist ← 1
41:         end if
42:         if $l == L$ OR ($L > 0$ AND $l_i > 0$)
43:             stopLinNonSep ← 1
44:         end if
45:         if stopDist == 1 AND stopLinNonSep == 1
46:             break the loop               ▷ Cluster $\mathbf{c}_i$ is in its final position.
47:         end if
48:     end for
49:     $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{c}_i$
50: end for
```

2.2.3 Higher dimensions

We plan to extend the presented data generator in the future to produce datasets of dimensionality higher than two. One will have to implement algorithm for n-dimensional α-shapes as described in [33] and adapt the routines for cluster scaling, translation, and rotation; other adjustments of the algorithm are trivial. In this thesis, the complexity of the high-dimensional problems is present in the GENE and REAL datasets that are presented in Chapter 3.

Algorithm 2

Data generator helper functions.

```
 1: function ClusterCreate($N$)
 2:     shape ← random element from the list shapes
 3:     $\mathbf{c}$ ← Generate $N$ data points that fill the shape using selected distribution.
 4:     Scale the cluster $\mathbf{c}$ by a random real factor on $[0, 1]$.
 5:     return $\mathbf{c}$
 6: end function
 7: function ClusterMove($\mathbf{c}, \mathbf{a}, \mathbf{b}$)
 8:     Move every point in the cluster $\mathbf{c}$ by the vector $\mathbf{b} - \mathbf{a}$.
 9:     return $\mathbf{c}$
10: end function
11: function ClusterRotate($\mathbf{c}, \mathbf{a}, \beta_{\max}$)
12:     $\beta$ ← random angle in the range $[0, \beta_{\max}]$.
13:     Rotate points in the cluster $\mathbf{c}$ around reference point $\mathbf{a}$ by the angle $\beta$.
14:     return $\mathbf{c}$
15: end function
16: function Bnd($\mathbf{c}$)
17:     $\mathbf{b}$ ← set of points on a $\alpha$-shape boundary of the cluster $\mathbf{c}$
18:     return $\mathbf{b}$
19: end function
```

2.3 `Complex2D` *dataset family*

Using the proposed data generator, we create a family of datasets named `Complex2D`. The purpose of this collection is to test how the distance between clusters and their non-linear entanglement influence the performance of cluster algorithms and validity indices as well. We defined three different scenarios S_1, S_2, and S_3, varying parameters L, $d_{\min}$, and K:

- S_1: $L = 0$, $d_{\min} = \{0.3, 0.4, 0.5\}$, $K = \{2, 3, 4, 6, 8, 10\}$,
- S_2: $L = 1$, $d_{\min} = \{0.08, 0.1, 0.15\}$, $K = \{2, 3, 4, 6, 8, 10\}$,

- S_3: $L = 2$, $d_{\min} = \{0.08, 0.1, 0.15\}$, $K = \{3, 4, 6, 8, 10\}$.

With these settings we graduate the difficulty of the generated datasets. S_1 represents easy cases where clusters are linearly separable and there is a relatively large distance between them. Scenario S_2 represents moderate-level difficulty with two clusters being linearly non-separable and with narrower gaps between clusters' borders. In the same manner, the S_3 is defined with L being 2. The values for $d_{\min}$ are determined experimentally in accordance with L – the bigger the L is, the lower the $d_{\min}$ has to be in order to ensure the convergence of the algorithm. We set the upper limit for the number of clusters K to 10, which results in only 50 data points in each cluster if the total number of data points is 500 as in our case. So, with even higher number of clusters, these would be poorly populated and thus without the desired shape.

The following features are in common of all the scenarios:

- $N = 500$,
- `distribution` = {uniform, truncated normal, mixed[4]},
- each configuration of the parameters is used 10 times,
- the default values of other settings.

Our generator enables an arbitrary distribution of data samples across clusters. Although datasets with imbalanced clusters present a kind of challenge to a clusterer [21, 34], we decided to generate a collection of datasets with equally sized clusters and rather give more importance on the complexity of cluster shapes and linear separability. However, the difficulty of imbalanced data is present in the `REAL` and `GENE` datasets, for instance in the datasets `fertility`, `voice_3`, `leukemia-1`, and `mesothelioma` to mention only few, where one cluster contains substantially more data samples than others. As already said, the data points in the `Complex2D` datasets are evenly distributed across K clusters using the following equations

$$n_i = \left\lfloor \frac{N}{K} \right\rfloor + r_i\,, \tag{2.10}$$

[4]Mixed distribution means that for each cluster the distribution of data points is chosen at random between uniform and truncated normal.

$$r_i = \begin{cases} 1, & \text{if } i \leq \left(N - K\left\lfloor \frac{N}{K} \right\rfloor\right) \\ 0, & \text{otherwise} \end{cases} . \tag{2.11}$$

We created one dataset for every possible configuration of the generator's parameters. For example, considering scenario S_1, we have 3 values for $d_{\min}$, 6 values for K, 3 values for `distribution` and 10 repetitions of them, resulting in $3 \cdot 6 \cdot 3 \cdot 10 = 540$ different datasets. The same is true for the scenario S_2. The scenario S_3 produces datasets with $K \geq 3$ clusters, while it is not meaningful to have only two clusters and force two pairs of clusters to be linearly non-separable. So, there are $3 \cdot 5 \cdot 3 \cdot 10 = 450$ datasets in total for S_3.

For illustration, in the Fig. 2.3 we plotted four examples of `Complex2D` datasets. The sub-plot a) depicts scenario S_1 with two clusters, $d_{\min} = 0.5$, $L = 0$, `distribution` = uniform; the dataset in the sub-plot b) was also generated by scenario S_1 and has six clusters, $d_{\min} = 0.3$, $L = 0$, `distribution` = uniform. In c) we can see a dataset from the scenario S_2 with four clusters, $d_{\min} = 0.1$, $L = 1$, `distribution` = truncated normal. Example d) is from the scenario S_3 with eight clusters, $d_{\min} = 0.08$, $L = 2$, `distribution` = mixed. Datasets are scaled proportionally to fit in the interval $[0, 1]$ in both dimensions.

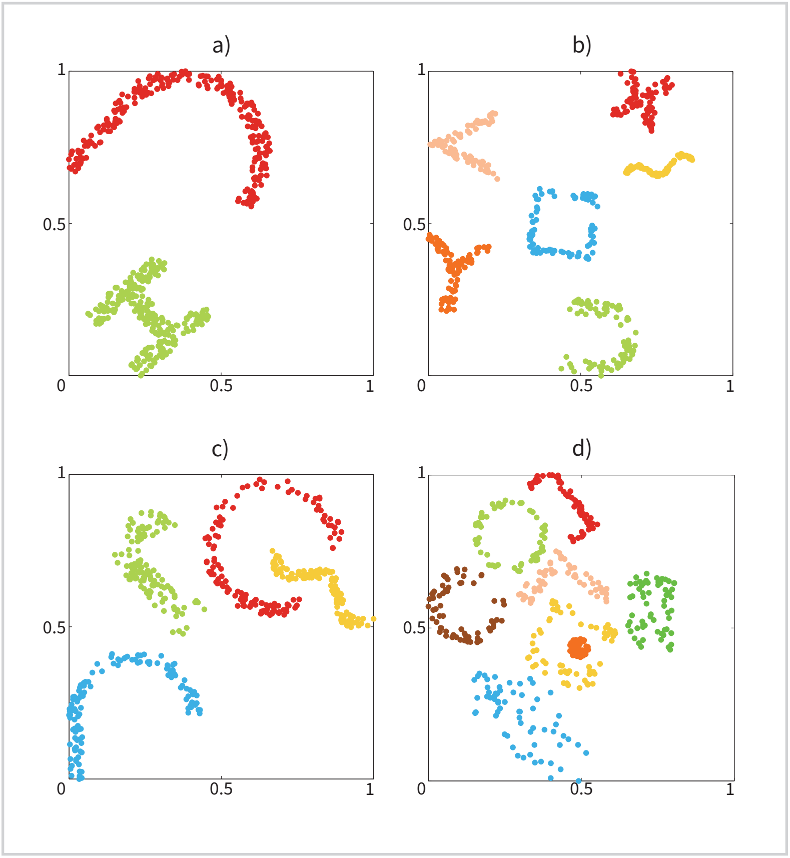


Figure 2.3

Examples of `Complex2D` datasets:

a) Scenario S_1, $K = 2$, $d_{\min} = 0.5$, $L = 0$, `distribution` = uniform;

b) Scenario S_1, $K = 6$, $d_{\min} = 0.3$, $L = 0$, `distribution` = uniform;

c) Scenario S_2, $K = 4$, $d_{\min} = 0.1$, $L = 1$, `distribution` = truncated normal;

d) Scenario S_3, $K = 8$, $d_{\min} = 0.08$, $L = 2$, `distribution` = mixed.

Cluster validation

3.1 Introduction

The essential, yet often neglected, step in the cluster analysis is a validation of the clustering results [35]. To assess the output of the clustering procedure, a wealth of cluster validity indices has been proposed so far [11, 36–38]. Generally, they are classified into two groups: internal and external cluster validity indices. The internal cluster validity indices (CVI) evaluate the given partition of the data by measuring the compactness and the separation of the clusters on a basis of some objective criteria, without any information about how the *true* or correct partition should look like. On the contrary, the external cluster validity indices (eCVI) validate the clustering result with a reference to the partition that is known to be the ground-truth. As such, eCVI offers more objective assessment of a given partition. However, in the real-world applications the ground-truth is usually not known beforehand, so one should use the internal indices to measure how well do the obtained partitions fit the input data. Besides, the majority of clustering algorithms require the expected number of clusters K in the data to be set in advance. Hence, multiple runs of an algorithm are executed for different values of K and an internal index is employed afterwards to pick out the best partition.

One possible classification of the CVIs is on optimization-like and difference-like indices [38]. The first ones are those, which select the best clustering solution or a partition with their maximum or minimum value. The difference-like indices consider the sequence of partitions with increasing number of clusters: the magnitude of difference between two consecutive partitions is a basis for detection of the best solution. In this thesis we address only the optimization-like CVIs.

One of the most used CVI was proposed by Dunn in 1973 [19] and since then some of its generalizations have been introduced, using different measures of the compactness and the separation between the clusters [39, 40]. In this chapter, we propose a new Dunn-like validity index based on the shortest paths between the data points considering the Gabriel graph on the data. Our index can be seen as a modification of generalized Dunn's index, proposed by Pal and Biswas [39], yet improving its ability to correctly identify good-quality partitions when arbitrary-shaped clusters are present in the data, which is still a challenge in the field [41]. To verify, whether an index has the ability of validating clusters of complex shapes, we developed a synthetic data generator that is able to produce clusters of highly irregular shapes and can control the degree of linear-separability between them (see Chapter 2). Then we experimentally compared the performance of 34 validation indices, along with the proposed one, on the synthetic as well as on real-world datasets of various types.

Let us first introduce the Gabriel graph and motivate its usage. Then we present more formally the generalization of the Dunn's index using the Gabriel graph and its proposed modification.

3.2 Gabriel graph

One of the most challenging problems for the internal indices is dealing with clusters of non-spherical shapes. Therefore, new approaches have emerged that are based on the graph theory concepts and are able to capture the real structure of the data more efficiently [39, 42]. Pal and Biswas used three types of graphs to impose a structure on the data, i.e. minimum spanning tree, relative neighbourhood graph, and Gabriel graph. Their results on various datasets show that the generalized Dunn's index based on the Gabriel graph [43] achieves the best performance. Furthermore, the connectivity properties of the Gabriel graph prove to be beneficial in terms of the cluster analysis as shown in [44]. These are the reasons why we adopted the Gabriel graph as the foundation of our research as well.

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ be a set of D-dimensional data points, $\mathbf{x}_i \in \mathbb{R}^D$. A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is an ordered pair, where $\mathbf{V} = \{v_1, v_2, \dots, v_N\}$ is a set of vertices and $\mathbf{E} = \{e_1, e_2, \dots, e_L\}$ is a set of edges between the vertices in $\mathbf{V}$. For each data point $\mathbf{x}_i$ there is a vertex v_i that is its abstraction in the graph $\mathbf{G}$, thus $|\mathbf{X}| = |\mathbf{V}| = N$. The proximity of the vertices $v_i, v_j \in \mathbf{V}$ is defined as the Euclidean distance between the corresponding pair of data points, i.e. $d_E(\mathbf{x}_i, \mathbf{x}_j)$. Let edge $e_q = \{v_i, v_j\}$ link the vertex v_i with the vertex v_j and let $\mathbf{G}$ be an undirected weighted graph – it means that the direction of an edge is neglected and that the edges are weighted by the Euclidean distance between the data points. Thus, the weight of the particular edge $e_q = \{v_i, v_j\}$ is computed as $w(e_q) = d_E(\mathbf{x}_i, \mathbf{x}_j)$.

The Gabriel graph is a graph, in which for any two vertices $v_i, v_j \in \mathbf{V}$ there is an edge $e_q = \{v_i, v_j\}$ between two vertices, if

$$d_E^2(\mathbf{x}_i, \mathbf{x}_j) < d_E^2(\mathbf{x}_i, \mathbf{x}_k) + d_E^2(\mathbf{x}_k, \mathbf{x}_j)\,, \tag{3.1}$$

$\forall k : v_k \in \mathbf{V}, k \neq i, k \neq j$. In other words, vertices v_i and v_j are connected, if there does not exist any other vertex v_k, such that its corresponding data point $\mathbf{x}_k$

would fall into the D-dimensional hypersphere with diameter $d_E(\mathbf{x}_i, \mathbf{x}_j)$ and its centre in $\mathbf{x}_i + (\mathbf{x}_j - \mathbf{x}_i)/2$. We illustrate this notion on an example of three two-dimensional data points in Fig. 3.1a). In the following sections, we use a demonstrative dataset LCA to visually explain different modifications of the Dunn's index. In Fig. 3.1b) we show the Gabriel graph on this dataset.

In order to compute the Gabriel graph with a greedy algorithm, we have to compute d_E, which has a single-pass time complexity of $O(D)$, with respect to the data dimensionality D. We need to evaluate the condition in Eq. (3.1) for all the triplets of the data points, so the overall time complexity of creating the Gabriel graph is $O(D \cdot N^3)$.

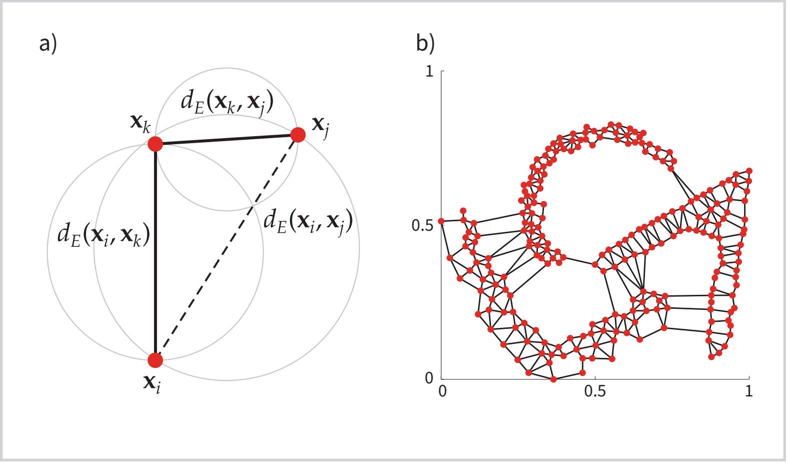

a)
$d_E(\mathbf{x}_k, \mathbf{x}_j)$
$\mathbf{x}_k$
$\mathbf{x}_j$
$d_E(\mathbf{x}_i, \mathbf{x}_k)$
$d_E(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbf{x}_i$
b)
1
0.5
0
0
0.5
1

Figure 3.1

a) A construction of the Gabriel graph on three data points depicted by red dots. Solid black line represents an edge between two vertices. b) Example of the Gabriel graph on a demonstrative LCA dataset.

3.3 Dunn's index and its generalization

Suppose we have partitioned dataset $\mathbf{X}$ into K clusters and have obtained the partition $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K\}$, such that $\mathbf{X} = \bigcup_{i=1}^{K} \mathbf{c}_i$ and $\mathbf{c}_i \cap \mathbf{c}_j = \emptyset$, $i \neq j, \mathbf{c}_i \neq \emptyset$. This means the partition $\mathbf{C}$ is *crisp* and each data point is a member of exactly one cluster. The Dunn's validity index [19] of the partition $\mathbf{C}$ is computed as

$$\mathrm{DN}(\mathbf{C}) = \frac{\min_{1 \le i,j \le K} \operatorname{dist}(\mathbf{c}_i, \mathbf{c}_j)}{\max_{1 \le k \le K} \operatorname{diam}(\mathbf{c}_k)}, \tag{3.2}$$

where $\mathrm{diam}(\mathbf{c}_k)$ represents the diameter of a cluster $\mathbf{c}_k$ and $\mathrm{dist}(\mathbf{c}_i, \mathbf{c}_j)$ the distance between two clusters $\mathbf{c}_i$ and $\mathbf{c}_j$. We compute those quantities as

$$\mathrm{diam}(\mathbf{c}_i) = \max_{\mathbf{x}_m, \mathbf{x}_n \in \mathbf{c}_i} d_E(\mathbf{x}_m, \mathbf{x}_n)\,, \tag{3.3}$$

$$\mathrm{dist}(\mathbf{c}_i, \mathbf{c}_j) = \min_{\mathbf{x}_m \in \mathbf{c}_i, \mathbf{x}_n \in \mathbf{c}_j, i \neq j} d_E(\mathbf{x}_m, \mathbf{x}_n)\,. \tag{3.4}$$

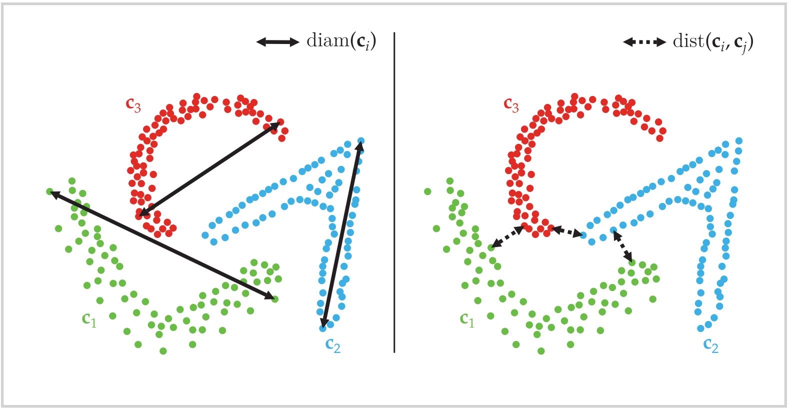


Figure 3.2

Diameters of clusters and distances between them as defined by the Dunn's index DN. Clusters are shown using different colours.

High value of $\mathrm{DN}(\mathbf{C})$ indicates compact and well separated clusters in the partition $\mathbf{C}$, thus we may compute partitions for different number of clusters K, or for any other parameter of the clustering algorithm, and consider the partition that maximizes the Dunn's index as the optimal solution.

To calculate the $\mathrm{DN}(\mathbf{C})$ we firstly have to compute the distances between all the data points. The between-point distances are then used to calculate the diameter of the cluster in Eq. (3.3) and the distance between clusters in Eq. (3.4). There are altogether $N(N-1)/2$ pairs of points, so the computation of the d_E distances requires $O(D \cdot N^2)$ time. Eq. (3.2) itself requires $O(K^2)$ time, so the complexity of Dunn's index is $O(D \cdot N^2 + K^2)$ and, considering that $K \leq N$, it reduces to $O(D \cdot N^2)$.

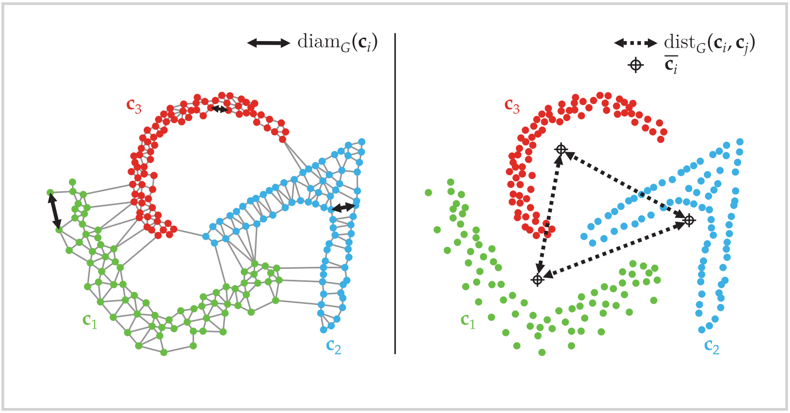
However, Pal and Bezdek argue that such definitions of the inter-cluster diameter and the between-cluster distance are too sensitive to noisy data points and are also inconvenient for the validation of non-spherical clusters [40]. As the answer, some indices, enhanced via various types of graphs, were provided; one of these indices uses the concept of the Gabriel graph and we refer to it as the generalized Dunn's index [39], abbreviated as DNg. The main idea about the generalization is to represent data points $\mathbf{x}_i \in \mathbf{X}$ with vertices $v_i \in \mathbf{V}$ in the Gabriel graph and to use the redefinition of the cluster diameter and the between-cluster distance. Pal and Biswas defined the diameter of the cluster $\mathbf{c}_i$ in the following way

$$\mathrm{diam}_G(\mathbf{c}_i) = \max_{1 \le q \le |\mathbf{E}_i|} \{e_q\},\ e_q \in \mathbf{E}_i\,, \tag{3.5}$$

where $\mathbf{E}_i$ denotes the set of edges in the Gabriel graph, such that every edge $e_q \in \mathbf{E}_i$ connects a pair of vertices that both belong to the cluster $\mathbf{c}_i$. Furthermore, they defined the distance between clusters $\mathbf{c}_i$ and $\mathbf{c}_j$ as the distance between the clusters' centroids

$$\mathrm{dist}_G(\mathbf{c}_i, \mathbf{c}_j) = d_E(\overline{\mathbf{c}_i}, \overline{\mathbf{c}_j})\,, \tag{3.6}$$

where $\overline{\mathbf{c}_i}$ denotes the mean value or the centroid of all the data points in the cluster $\mathbf{c}_i$. We illustrate Eq. (3.5) and Eq. (3.6) in Fig. 3.3.



$\mathrm{diam}_G(\mathbf{c}_i)$
$\mathbf{c}_3$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathrm{dist}_G(\mathbf{c}_i, \mathbf{c}_j)$
$\overline{\mathbf{c}_i}$
$\mathbf{c}_3$
$\mathbf{c}_1$
$\mathbf{c}_2$

Figure 3.3

Diameters of clusters and distances between them as defined by the generalized Dunn's index DNg. Grey lines are the edges of the Gabriel graph. Clusters are shown using different colours.

The generalized Dunn's index DNg of the partition $\mathbf{C}$ is calculated in a similar way as the original Dunn's index

$$\mathrm{DNg}(\mathbf{C}) = \frac{\min_{1 \le i,j \le K} \mathrm{dist}_G(\mathbf{c}_i, \mathbf{c}_j)}{\max_{1 \le k \le K} \mathrm{diam}_G(\mathbf{c}_k)}\,, \tag{3.7}$$

where the higher value of DNg($\mathbf{C}$) indicates better partition. To find the optimal partition $\mathbf{C}$, we search for the maximum of DNg($\mathbf{C}$). Authors demonstrated a good performance of the DNg index for both the structural or chain-like clusters and the spherical clusters. They also argued that their proposed index is more resistant to the noise, although this hypothesis is not explicitly proven in the paper [39].

The DNg index is more expensive to compute than the DN index due to the construction of the Gabriel graph, which takes $O(D \cdot N^3)$ time. Considering that there are at most $3N$ edges in the Gabriel graph [44], Eq. (3.5) can be computed in $O(N)$ time for all the clusters. The computation of the distances between all the pairs of clusters requires additional $O(D \cdot K^2)$ time, thus it all sums up to $O(D \cdot N^3 + D \cdot K^2 + N)$. Provided that $K \leq N$, it finally reduces to the complexity of $O(D \cdot N^3)$.

3.4 The proposed modified Dunn's index

Motivated by promising results of the generalized Dunn's index, we introduce its improvement as yet another modification of the way in which the diameter of cluster and the distance between clusters are computed. The reader should note that the proposed variant of the Dunn's index, abbreviated as DNs, is also a generalization of the original Dunn's index DN, but for the sake of clarity we refer to it as the modified Dunn's index to avoid a confusion with the DNg index.

The main idea of the proposed approach is to define the distance between a pair of data points in the terms of the shortest path between them in the Gabriel graph. We designed this definition of the distance to better describe the cluster's shape and its inner structure. In order to formalize the proposed distance, let us first introduce some essential definitions. As before, let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be the Gabriel graph built on the dataset $\mathbf{X}$. A path p between the vertices v_i and v_j is a sequence of vertices, such that there is an edge between each of the two successive vertices. A path with no repeated vertices is called a simple path and let l_p be the number of edges on the simple path p. With e_r^p, $r = 1, \dots, l_p$ we denote an edge on a path p. Suppose there are P possible paths between a pair of vertices $v_i, v_j \in \mathbf{V}$. We propose that the distance between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as the sum of edge weights along the shortest path between the vertices v_i and v_j

$$d_S(\mathbf{x}_i, \mathbf{x}_j) = \min_{1 \le p \le P} \left\{ \sum_{r=1}^{l_p} w(e_r^p) \right\} , \tag{3.8}$$

where $w(e_r^p)$ is the weight of the edge on the path p between two successive vertices v_i and v_j and it equals $d_E(\mathbf{x}_i, \mathbf{x}_j)$. To find the shortest paths between all pairs of vertices in $\mathbf{V}$, we employ a well-known Johnson's algorithm, which can be computed in $O(N^2 \log N + N^2)$ time assuming that $|\mathbf{E}| = O(N)$ [45]. In the Gabriel graph, there always exists a path between any of two vertices (for proof, see [44]), so $d_S(\cdot, \cdot)$ is a non-negative real number for all the pairs of the data in $\mathbf{X}$.

Now, the definitions of the cluster diameter in Eq. (3.3) and the between-clusters distance in Eq. (3.4) are altered slightly by a substitution of the Euclidean distance d_E with the distance d_S

$$\mathrm{diam}_S(\mathbf{c}_i) = \max_{\mathbf{x}_m, \mathbf{x}_n \in \mathbf{c}_i} d_S(\mathbf{x}_m, \mathbf{x}_n) , \tag{3.9}$$

$$\mathrm{dist}_S(\mathbf{c}_i, \mathbf{c}_j) = \min_{\mathbf{x}_m \in \mathbf{c}_i,\, \mathbf{x}_n \in \mathbf{c}_j,\, i \neq j} d_S(\mathbf{x}_m, \mathbf{x}_n) . \tag{3.10}$$

Please note, that the computation of cluster's diameter in Eq. (3.9) involves only the data points in that cluster. So, instead of using the graph on all the data points, we create a K smaller graphs on the points from each individual cluster. To get the distances between the clusters as defined in Eq. (3.10), we need the graph on the whole dataset $\mathbf{X}$.

We graphically demonstrate the proposed redefinitions of diameter and distance in Fig. 3.4. On the left hand side of the figure, we see that the cluster's diameter precisely follows the cluster's shape and provides with the greater detail of abstraction than the definitions of diam in Eq. (3.3), where the diameter of a cluster is actually a diameter of a hypersphere passing through the most distant points of a cluster. As such, it is more convenient for clusters of globular shape. The diameter diam_G in Eq. (3.5) is rather a measure of cluster density as it increases if the edges between cluster members are longer, and vice-versa. Considering the distance between clusters, our proposal does not differ from the original Dunn's definition so radically. The distances are following the edges in the Gabriel graph too as it represents the topological abstraction of a given dataset. Thus, we expect our proposals would result in a meaningful description of a partition under evaluation.

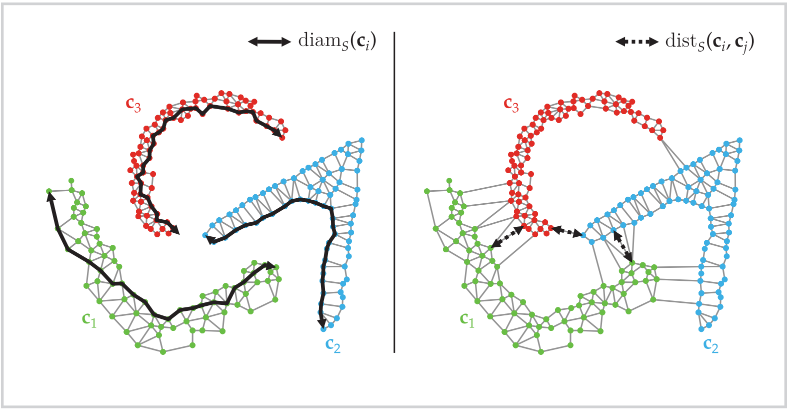

$\text{diam}_S(\mathbf{c}_i)$
$\text{dist}_S(\mathbf{c}_i, \mathbf{c}_j)$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_3$

Figure 3.4

Diameters of clusters and distances between them as defined by the modified Dunn's index DNs. Grey lines are the edges of the Gabriel graph. Clusters are shown using different colours.

Some cluster validity indices, for example the index I [46] and the Score function [47], utilize yet another principle to distinguish between partitions of different quality – they penalize partitions with large number of clusters. In a few following paragraphs we show the benefits of penalization and why we adopt this idea when designing our proposal.

The reasoning behind the penalization is that in a typical situation of cluster analysis we expect a lot fewer clusters in a data than there are data points, i.e. $K \ll N$. Actually, this is the whole idea of clustering: to group or organize data in meaningful subsets with similar features. If there is too many clusters discovered, a data miner could doubt whether his or her data contain meaningful structure. Therefore, without prior knowledge about the data we have, we usually cluster them repeatedly with varying K from 2 to some arbitrary upper limit $K_{\max}$. In the literature, some researchers just pick an arbitrary value higher than expected number of clusters, e.g. $K_{\max} = 10$ [41] or $K_{\max} = 12$ [48]. Yet others use the rule-of-thumb $K_{\max} = \lceil \sqrt{N} \rceil$ [11, 37, 40, 49]. Then we validate the resulting partitions with one or more validity indices and pick the solution that optimizes the indices.

However, the majority of cluster validation indices' definitions contain a notion of cluster compactness. For example, considering the Dunn-like indices we called it a cluster's diameter in Eq. (3.3), (3.5), and (3.9). It is based on a closeness between data points in the same cluster. To optimize the index in Eq. (3.2), the diameter must be minimized. But, this optimum is reached when every data point falls into its own cluster and the diameter equals 0. This can be seen in Fig. 3.5, where we demonstrate the behaviour of maximum diameter diam_S as a green dotted line and minimum distance dist_S as a brown dotted line when clustering the dataset LCA with single-linkage[1] algorithm [50]. We ran the clusterer for $K = \{2, 3, \dots, N\}$ and we can see how both quantities decrease when K increases.

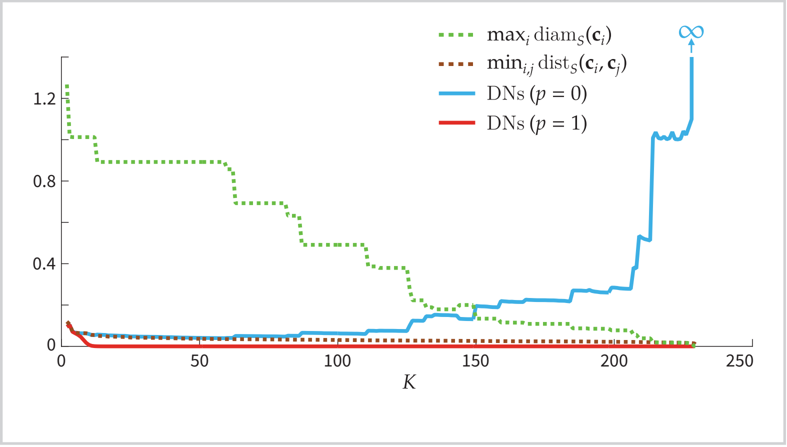


Figure 3.5

The LCA dataset is clustered with single-linkage algorithm into $K = \{2, \dots, N\}$ clusters. The figure shows the behaviour of a maximum cluster diameter and minimum between cluster distance as well as the values of the proposed index without and with penalization term.

So, the term of compactness in the equation tends to prefer the solutions where K approaches N. This is why we introduce the penalty function f that decreases with increasing K. Among many possible functions we chose a logistic function, defined as

$$f(K) = \frac{1}{1 + p \cdot e^{p(K-K_0)}}, \qquad (3.11)$$

where p and K_0 are the parameters a user can control. The parameter $p \in [0, \infty)$ controls the shape or steepness of the sigmoid curve and can be understood as an amount of penal – if $p = 0$, there is no penal. The parameter K_0 is the sigmoid midpoint, where the curve reaches the value 0.5. The main reason we chose logistic function is its ability to leave the smaller values of K intact and penalize only the larger

[1]We describe the single-linkage clustering algorithm together with average-linkage and complete-linkage variants in Chapter 5.

ones. With K_0 we are able to tell, which number of clusters is still plausible for our data and where the penalization may start. If we do not have any presumptions about the data, we can use the default values of the parameters: in our experiments we set $p = 1$ and $K_0 = \lceil \sqrt{N} \rceil / 2$. The graphs in Fig. 3.6 depict six variants of a penalty function f, where K_0 is fixed to 12 and p varies from 0 to 1.

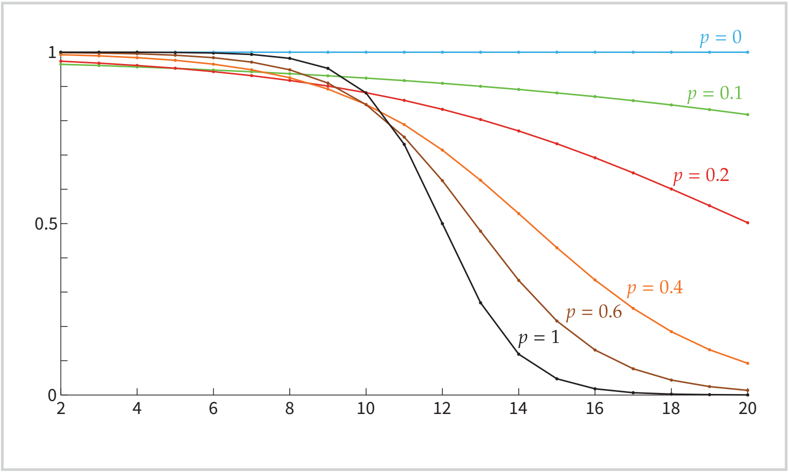


Figure 3.6

Logistic penalty function f for $K = 2, \dots, 20$, $K_0 = 12$ and six variants of p.

Now, we define the modified Dunn's index as

$$\mathrm{DNs}(\mathbf{C}) = f(K) \cdot \frac{\min_{1 \le i,j \le K} \mathrm{dist}_S(\mathbf{c}_i, \mathbf{c}_j)}{\max_{1 \le k \le K} \mathrm{diam}_S(\mathbf{c}_k)} \,. \tag{3.12}$$

Its behaviour, when validating the partition $\mathbf{C}$ is exactly the same as with the DN or the DNg index – the larger the value of DNs($\mathbf{C}$) is, the better the partition $\mathbf{C}$ is considered to be, relatively to other partitions in a comparison. Thus, the maximum diameter of the clusters in the partition $\mathbf{C}$ should be minimized and the minimum distance between any of two clusters should be maximized in order to optimize the DNs index. If we return once more to Fig. 3.5 – with solid lines we plot the values of our proposed index DNs. We apply it without and with the penalization, which results in blue and red solid line, respectively. With increasing K the difference between both options is obvious: without penalization, the solutions with large number of clusters are preferred, whereas the penal for large K inhibits this negative effect.

At this point, let us briefly compare the discussed Dunn-like CVIs on the `LCA` dataset. We cluster the data using the single-linkage algorithm for $K = \{2, 3, \dots, K_{\max}\}$, where $K_{\max} = \lceil \sqrt{N} \rceil = \lceil \sqrt{230} \rceil = 16$. The resulting 15 partitions are then evaluated by the DN, DNg, and DNs indices. The latter is executed with parameter p set to 0, i.e. no penal, and 1. In Fig. 3.7 we see the values of the indices as blue dots and the optimums are highlighted with red circle. As it is clear from the top panel of the figure, index DN chooses the partition with two clusters as the best one, whereas the index DNg points to the four-clusters solution. The proposed index DNs without penalization as well as with $p = 1$ picks the partition with three clusters as the optimal one. The mid-point of a penalizing function f is set to $K_0 = K_{\max}/2 = 8$. In this case, a guess of DNs matches the assumed structure of the dataset perfectly. Please note, that this kind of evaluation, where the predicted number of clusters should match the *true* one, is not the only one. We discuss different evaluation methodologies later in Section 3.6.2.

When analysing the time complexity of the DNs index, we consider three computational steps. Firstly, the Gabriel graph is built on the whole dataset in $O(D \cdot N^3)$ time and for each cluster separately in $O(D \cdot \sum_{i=1}^{K} |\mathbf{c}_i|^3)$ time. Asymptomatically, it equals the complexity of $O(D \cdot N^3)$. Secondly, Johnson's algorithm is applied, which takes $O(N^2 \log N + N^2)$ time. Finally, Eq. (3.12) requires $O(K^2)$ time. Hence, the complexity of the DNs index asymptotically equals $O(D \cdot N^3)$. The time complexity of the DNs index is therefore higher than that of the DN index, i.e. $O(D \cdot N^2)$, and grows as fast as that of the DNg index, i.e. $O(D \cdot N^3)$.

3.5 *Definitions of cluster validity indices*

In this section, we provide a brief overview of the CVIs that are used in a comparison with the proposed index DNs. From the literature, we selected those that behave as an optimization function, which has to be either minimized or maximized in order to get the best solution among partitions. We say an index is min-like, denoted as [min.], if the optimal partition minimizes the index's value. Similarly, an index is max-like, denoted as [max.], if optimal partition maximizes its value. In the following, for the sake of clarity, we use this format of subsection titles: `Index's full name (abbreviation) [min./max.]`.

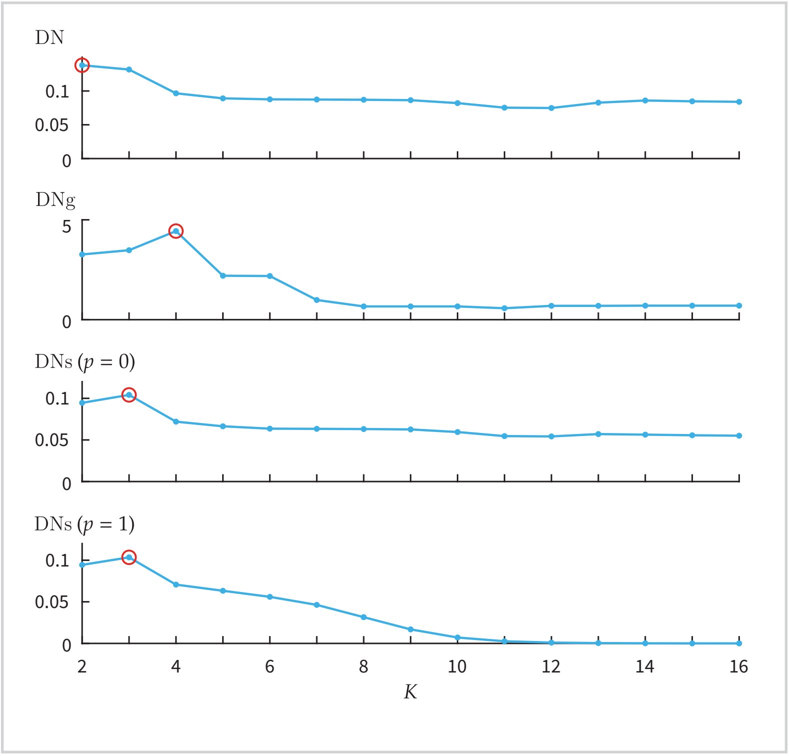


Figure 3.7

The values of the indices DN, DNg, DNs with $p = 0$, and DNs with $p = 1$ when partitioning the LCA dataset using single-linkage algorithm with various K.

Calinski-Harabasz (CH) [max.]

Calinski-Harabasz index [51] is computed as normalized ratio between separation and compactness within clusters. Separation is measured as distance between the cluster centroid and the centroid of data set. The compactness of cluster is based on distance between data points and the cluster centroid. The larger the value of $CH(\mathbf{C})$, the better the partition $\mathbf{C}$.

$$\mathrm{CH}(\mathbf{C}) = \frac{N-K}{K-1} \frac{\sum_{\mathbf{c}_k \in \mathbf{C}} |\mathbf{c}_k| \cdot d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{X}})}{\sum_{\mathbf{c}_k \in \mathbf{C}} \sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k})} . \tag{3.13}$$

C-index (CI) [min.]

Huber and Levin [52] defined a criterion based on sum of distances within clusters Θ, normalized by worst-case (Θ_{max}) and best-case (Θ_{min}) scenarios as

$$\mathrm{CI}(\mathbf{C}) = \frac{\Theta - \Theta_{min}}{\Theta_{max} - \Theta_{min}}, \tag{3.14}$$

where

$$\Theta = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} q_{i,j} d_E(\mathbf{x}_i, \mathbf{x}_j) \tag{3.15}$$

and

$$q_{i,j} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are in the same cluster,} \\ 0, & \text{otherwise.} \end{cases} . \tag{3.16}$$

Θ_{min} is obtained by sorting values of $d_E(\mathbf{x}_i, \mathbf{x}_j)$ in an ascending order and then summing up the first N_w distances, where N_w is the number of all pairs of data points that share the same cluster, i.e. $N_w = \sum_{\mathbf{c}_k \in \mathbf{C}} \binom{|\mathbf{c}_k|}{2}$. We compute Θ_{max} in the similar manner summing up the last N_w values of the sorted distances between the data points.

Connectivity index (CON) [min.]

The connectivity index [35] evaluates the degree to which neighbouring data points have been placed in the same cluster. It is defined as

$$\mathrm{CON}(\mathbf{C}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} q_{i,\mathrm{nn}_i(j)} \right), \tag{3.17}$$

$$q_{i,\mathrm{nn}_i(j)} = \begin{cases} \frac{1}{j}, & \text{if } \nexists \mathbf{c}_k : \mathbf{x}_i \in \mathbf{c}_k \wedge \mathrm{nn}_i(j) \in \mathbf{c}_k \\ 0, & \text{otherwise} \end{cases}, \tag{3.18}$$

where $\mathrm{nn}_i(j)$ is the jth nearest neighbour of the data point $\mathbf{x}_i$, and L is a parameter determining the number of neighbours that contribute to the index CON. We followed the procedure in [10] and set the parameter L to the value of 5 for all the experiments.

Davies-Bouldin index (DB) [min.]

Davies and Bouldin [53] proposed an index based on a ratio of within-cluster and between-cluster distances. The within-cluster distance is an average distance between the data points in a cluster and its centroid. The between-cluster distance is defined as a distance between centroids.

$$\mathrm{DB}(\mathbf{C}) = \frac{1}{K} \sum_{\mathbf{c}_k \in \mathbf{C}} \max_{\mathbf{c}_l \in \mathbf{C}, k \neq l} \left\{ \frac{S(\mathbf{c}_k) + S(\mathbf{c}_l)}{d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{c}_l})} \right\} , \tag{3.19}$$

where

$$S(\mathbf{c}_k) = \frac{1}{|\mathbf{c}_k|} \sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k}) \,. \tag{3.20}$$

Davies-Bouldin index - modified (DBmod) [min.]

Kim and Ramakrishna [54] proposed a modification of DB index by a redefinition of Eq. (3.19)

$$\mathrm{DBmod}(\mathbf{C}) = \frac{1}{K} \sum_{\mathbf{c}_k \in \mathbf{C}} \frac{\max_{\mathbf{c}_l \in \mathbf{C}, k \neq l} \{S(\mathbf{c}_k) + S(\mathbf{c}_l)\}}{\min_{\mathbf{c}_j \in \mathbf{C}, k \neq j} d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{c}_j})} \,. \tag{3.21}$$

Dunn's index (DN) [max.]

The Dunn's validity index [19] is computed as in Eq. (3.2).

Generalized Dunn's index based on a graph theory (DNg) [max.]

The generalized Dunn's validity index based on the graphs [39] is computed as in Eq. (3.7).

Generalized Dunn's Indices (GDN[i,j]) [max.]

Another family of Dunn-like indices was proposed by Bezdek and Pal [40]. They modified the way the distance between clusters and within cluster diameter is defined; equations Eq. (3.3) and Eq. (3.4) have been altered, resulting in five distance functions[2]:

[2]Note that dist_1 and diam_1 refer to the definitions of the *original* DN index in equations Eq. (3.3) and Eq. (3.4).

$$\mathrm{dist}_2(\mathbf{c}_k, \mathbf{c}_l) = \max_{\mathbf{x}_m \in \mathbf{c}_k, \mathbf{x}_n \in \mathbf{c}_l} d_E(\mathbf{x}_m, \mathbf{x}_n)\,, \tag{3.22}$$

$$\mathrm{dist}_3(\mathbf{c}_k, \mathbf{c}_l) = \frac{1}{|\mathbf{c}_k| \cdot |\mathbf{c}_l|} \sum_{\mathbf{x}_m \in \mathbf{c}_k, \mathbf{x}_n \in \mathbf{c}_l} d_E(\mathbf{x}_m, \mathbf{x}_n)\,, \tag{3.23}$$

$$\mathrm{dist}_4(\mathbf{c}_k, \mathbf{c}_l) = d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{c}_l})\,, \tag{3.24}$$

$$\mathrm{dist}_5(\mathbf{c}_k, \mathbf{c}_l) = \frac{1}{|\mathbf{c}_k| + |\mathbf{c}_l|} \left(\sum_{\mathbf{x}_m \in \mathbf{c}_k} d_E(\mathbf{x}_m, \overline{\mathbf{c}_k}) + \sum_{\mathbf{x}_n \in \mathbf{c}_l} d_E(\mathbf{x}_n, \overline{\mathbf{c}_l}) \right)\,, \tag{3.25}$$

$$\mathrm{dist}_6(\mathbf{c}_k, \mathbf{c}_l) = \max \left\{ \max_{\mathbf{x}_m \in \mathbf{c}_k} \min_{\mathbf{x}_n \in \mathbf{c}_l} \{d_E(\mathbf{x}_m, \mathbf{x}_n)\}, \max_{\mathbf{x}_n \in \mathbf{c}_l} \min_{\mathbf{x}_m \in \mathbf{c}_k} \{d_E(\mathbf{x}_m, \mathbf{x}_n)\} \right\}\,, \tag{3.26}$$

and two diameter functions

$$\mathrm{diam}_2(\mathbf{c}_k) = \frac{1}{|\mathbf{c}_k|\,(|\mathbf{c}_k| - 1)} \sum_{\mathbf{x}_m, \mathbf{x}_n \in \mathbf{c}_k, \mathbf{x}_m \neq \mathbf{x}_n} d_E(\mathbf{x}_m, \mathbf{x}_n)\,, \tag{3.27}$$

$$\mathrm{diam}_3(\mathbf{c}_k) = \frac{2}{|\mathbf{c}_k|} \sum_{\mathbf{x}_m \in \mathbf{c}_k} d_E(\mathbf{x}_m, \overline{\mathbf{c}_k})\,. \tag{3.28}$$

We write the generalized Dunn's indices in a form GDN[i,j], where i corresponds to the six distance functions and j to the three diameter functions. So, GDN[1,1] equals the original Dunn's index DN, other 17 combinations are its modifications.

In our experiments, presented in Section 3.6, we left out the three GDN indices, which have dist_5 as between-clusters distance function. The reason is that when we compared the implementations of the open-source projects `clusterCrit`[3] and `clv`[4], there were discrepancies in the results when applying GDN[5,1], GDN[5,2], and GDN[5,3] indices.

Homogeneity (HOM, HOMmin) [max.]

In his doctoral thesis, Roded Sharan [55] defined homogeneity (HOM) of a cluster as a function of similarity between cluster members. Average homogeneity of a partition

[3]Available at `https://cran.r-project.org/web/packages/clusterCrit`.

[4]Available at `https://cran.r-project.org/web/packages/clv`.

C is

$$\mathrm{HOM}(\mathbf{C}) = \sum_{\mathbf{c}_k \in \mathbf{C}} \frac{\sum_{\mathbf{x}_m, \mathbf{x}_n \in \mathbf{c}_k} \mathrm{sim}(\mathbf{x}_m, \mathbf{x}_n)}{\binom{|\mathbf{c}_k|}{2}} . \tag{3.29}$$

Here, a similarity is defined as inverse of the Euclidean distance

$$\mathrm{sim}(\mathbf{x}_m, \mathbf{x}_n) = \frac{1}{1 + d_E(\mathbf{x}_m, \mathbf{x}_n)} . \tag{3.30}$$

Worst-case scenario results in a minimum cluster homogeneity as

$$\mathrm{HOMmin}(\mathbf{C}) = \min_{\mathbf{c}_k \in \mathbf{C}} \left\{ \frac{\sum_{\mathbf{x}_m, \mathbf{x}_n \in \mathbf{c}_k} \mathrm{sim}(\mathbf{x}_m, \mathbf{x}_n)}{\binom{|\mathbf{c}_k|}{2}} \right\} . \tag{3.31}$$

I-index (I) [max.]

Maulik and Bandyopadhyay proposed the index I [46] that is defined as

$$\mathrm{I}(\mathbf{C}) = \left(\frac{\sum_{\mathbf{x}_m \in \mathbf{X}} d_E(\mathbf{x}_m, \overline{\mathbf{X}}) \cdot \max_{\mathbf{c}_k, \mathbf{c}_l \in \mathbf{C}} d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{c}_l})}{K \cdot \sum_{\mathbf{c}_k \in \mathbf{C}} \sum_{\mathbf{x}_m \in \mathbf{c}_k} d_E(\mathbf{x}_m, \overline{\mathbf{c}_k})} \right)^p , \tag{3.32}$$

where p is a parameter and is 2 by default[5].

S_Dbw index (SDBW) [min.]

Halkidi and Vazirgiannis proposed the validity index S_Dbw that measures a partition quality by the variance or the scatter of clusters and by the density between clusters [57]. The overview equation of the index is

$$\mathrm{SDBW}(\mathbf{C}) = \mathrm{scat}(\mathbf{C}) + \mathrm{dens}(\mathbf{C}) , \tag{3.33}$$

where the scatter of the partition **C** is defined as

$$\mathrm{scat}(\mathbf{C}) = \frac{1}{K} \sum_{\mathbf{c}_k \in \mathbf{C}} \frac{||\sigma(\mathbf{c}_k)||}{||\sigma(\mathbf{X})||} , \tag{3.34}$$

$$\sigma(\mathbf{S}) = \frac{1}{|\mathbf{S}|} \sum_{\mathbf{x}_i \in \mathbf{S}} (\mathbf{x}_i - \overline{\mathbf{S}})^2 , \tag{3.35}$$

[5]When $p = 2$, index I is also called the PBM index [56].

and

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}\,. \tag{3.36}$$

The density of the partition $\mathbf{C}$ depends on the density within and between clusters and is formulated as

$$\mathrm{dens}(\mathbf{C}) = \frac{1}{K(K-1)} \sum_{\substack{\mathbf{c}_k,\mathbf{c}_l \in \mathbf{C} \\ k \neq l}} \frac{\mathrm{dens}_B(\mathbf{c}_k, \mathbf{c}_l)}{\max\left\{\mathrm{dens}_W(\mathbf{c}_k), \mathrm{dens}_W(\mathbf{c}_l)\right\}}\,, \tag{3.37}$$

where

$$\mathrm{dens}_W(\mathbf{c}_k) = \sum_{\mathbf{x}_i \in \mathbf{c}_k} f(\mathbf{x}_i, \overline{\mathbf{c}_k})\,, \tag{3.38}$$

$$\mathrm{dens}_B(\mathbf{c}_k, \mathbf{c}_l) = \sum_{\mathbf{x}_i \in \mathbf{c}_k \cup \mathbf{c}_l} f(\mathbf{x}_i, \frac{\overline{\mathbf{c}_k} + \overline{\mathbf{c}_l}}{2})\,, \tag{3.39}$$

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} 0, & \text{if } d_E(\mathbf{x}, \mathbf{u}) > \mathrm{stdev}(\mathbf{C}) \\ 1, & \text{otherwise} \end{cases}\,, \tag{3.40}$$

and

$$\mathrm{stdev}(\mathbf{C}) = \frac{1}{K}\sqrt{\sum_{\mathbf{c}_k \in \mathbf{C}} \|\sigma(\mathbf{c}_k)\|}\,. \tag{3.41}$$

Separation (SEP, SEPmax) [min.]

Besides homogeneity HOM, Sharan also defined the separation SEP of a clustering as a function of similarity between members of different clusters [55]. Index SEP is an average separation of a partition $\mathbf{C}$

$$\mathrm{SEP}(\mathbf{C}) = \frac{1}{\binom{N}{2} - \sum_{\mathbf{c}_k \in \mathbf{C}} \binom{|\mathbf{c}_k|}{2}} \sum_{\substack{\mathbf{c}_k,\mathbf{c}_l \in \mathbf{C} \\ k \neq l}} \left(\sum_{\substack{\mathbf{x}_m \in \mathbf{c}_k \\ \mathbf{x}_n \in \mathbf{c}_l}} \mathrm{sim}(\mathbf{x}_m, \mathbf{x}_n) \right)\,, \tag{3.42}$$

where the similarity $\mathrm{sim}(\cdot,\cdot)$ is defined as in Eq. (3.30).

Worst-case scenario results in an index of maximum cluster separation

$$\mathrm{SEPmax}(\mathbf{C}) = \max_{\mathbf{c}_k,\mathbf{c}_l \in \mathbf{C}} \left\{ \frac{\sum_{\mathbf{x}_m \in \mathbf{c}_k, \mathbf{x}_n \in \mathbf{c}_l} \mathrm{sim}(\mathbf{x}_m, \mathbf{x}_n)}{|\mathbf{c}_k| \cdot |\mathbf{c}_l|} \right\}\,. \tag{3.43}$$

Score function (SF) [max.]

Saitta et al. proposed the index named Score function [47] that is defined as

$$\mathrm{SF}(\mathbf{C}) = 1 - \frac{1}{e^{e^{\mathrm{bcd}(\mathbf{C})-\mathrm{wcd}(\mathbf{C})}}} \,, \tag{3.44}$$

where the between class distance is defined as

$$\mathrm{bcd}(\mathbf{C}) = \frac{1}{N \cdot K} \sum_{\mathbf{c}_k \in \mathbf{C}} |\mathbf{c}_k| \cdot d_E(\overline{\mathbf{c}_k}, \overline{\mathbf{X}})^2 \tag{3.45}$$

and the within class distance as

$$\mathrm{wcd}(\mathbf{C}) = \frac{1}{K} \sum_{\mathbf{c}_k \in \mathbf{C}} \sqrt{\frac{1}{|\mathbf{c}_k|} \sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k})^2} \,. \tag{3.46}$$

Silhouette index (SIL) [max.]

Rousseeuw [58] proposed a silhouette measure of a data point that is based on a comparison of a compactness, measured as an average pair-wise distance to all the other cluster members, and a separation, computed as an average distance to all the data points in the nearest cluster. We compute the overall partition's performance, i.e. the silhouette index, as an average across all the data points as

$$\mathrm{SIL}(\mathbf{C}) = \frac{1}{N} \sum_{\mathbf{c}_k \in \mathbf{C}} \sum_{\mathbf{x}_i \in \mathbf{c}_k} \frac{b(\mathbf{x}_i, \mathbf{C}) - a(\mathbf{x}_i, \mathbf{c}_k)}{\max\{a(\mathbf{x}_i, \mathbf{c}_k), b(\mathbf{x}_i, \mathbf{C})\}} \,, \tag{3.47}$$

where

$$a(\mathbf{x}_i, \mathbf{c}_k) = \frac{1}{|\mathbf{c}_k|} \sum_{\mathbf{x}_j \in \mathbf{c}_k} d_E(\mathbf{x}_i, \mathbf{x}_j) \,, \tag{3.48}$$

$$b(\mathbf{x}_i, \mathbf{C}) = \min_{\mathbf{c}_l \in \mathbf{C}, \mathbf{x}_i \notin \mathbf{c}_l} \left\{ \frac{1}{|\mathbf{c}_l|} \sum_{\mathbf{x}_j \in \mathbf{c}_l} d_E(\mathbf{x}_i, \mathbf{x}_j) \right\} . \tag{3.49}$$

Simple structure index (SSI) [max.]

The Simple structure index [59] multiplicatively combines three features of a partition: the maximum difference of each variable between the clusters, the sizes of the most contrasting clusters and the deviation of a variable in the cluster centroids compared to its overall mean. Finally, the output value is normalized to interval $[0, 1]$.

Let's compute clusters' centroids, i.e. the means of their members, and put them into a matrix $\mathbf{A}$ of size $K \times D$ with clusters' centroids in its rows

$$\mathbf{A} = \begin{bmatrix} \overline{\mathbf{c}_1} \\ \overline{\mathbf{c}_2} \\ \vdots \\ \overline{\mathbf{c}_K} \end{bmatrix} = \begin{bmatrix} a_1^{(1)} & a_1^{(2)} & \cdots & a_1^{(D)} \\ a_2^{(1)} & a_2^{(2)} & \cdots & a_2^{(D)} \\ \vdots & \vdots & \ddots & \vdots \\ a_K^{(1)} & a_K^{(2)} & \cdots & a_K^{(D)} \end{bmatrix} . \tag{3.50}$$

With positive integers i and j we denote matrix's row and column index, respectively, so $1 \leq i \leq K$ and $1 \leq j \leq D$. Now, we define the maximum and the minimum of centroids variables as vectors

$$A_{\text{MAX}}^{(j)} = \max_i a_i^{(j)} , \tag{3.51}$$

$$A_{\text{MIN}}^{(j)} = \min_i a_i^{(j)} , \tag{3.52}$$

and the corresponding clusters indices, where the centroids variables reach their extremes

$$I_{\text{MAX}}^{(j)} = \arg\max_i a_i^{(j)} , \tag{3.53}$$

$$I_{\text{MIN}}^{(j)} = \arg\min_i a_i^{(j)} . \tag{3.54}$$

The clusters' sizes, i.e. the number of data points in the clusters, are defined as a list $\mathbf{S} = \{s_i : s_i = |\mathbf{c}_i|\} = \{|\mathbf{c}_1|, |\mathbf{c}_2|, \dots, |\mathbf{c}_K|\}$. Next, we pick the sizes, where centroids variables reach their extremes and put them into vectors

$$S_{\text{MAX}}^{(j)} = s_{I_{\text{MAX}}^{(j)}} , \tag{3.55}$$

$$S_{\text{MIN}}^{(j)} = s_{I_{\text{MIN}}^{(j)}} . \tag{3.56}$$

The Simple structure index is defined as follows

$$\text{SSI}(\mathbf{C}) = \frac{\sum_{j=1}^{D} r^{(j)} \cdot e^{-d^{(j)}} \cdot \sqrt{S_{\text{MAX}}^{(j)} \cdot S_{\text{MIN}}^{(j)}}}{u} , \tag{3.57}$$

where

$$
\begin{aligned}
r^{(j)} &= A^{(j)}_{\mathrm{MAX}} - A^{(j)}_{\mathrm{MIN}}\,, && (3.58)\\
d^{(j)} &= \left| \frac{\sum_{i=1}^{K} a_i^{(j)}}{K} - \frac{\sum_{i=1}^{K}\sum_{j=1}^{D} a_i^{(j)}}{K \cdot D} \right|, && (3.59)\\
u &= D \cdot \max\left\{ \max_j \left\{ S^{(j)}_{\mathrm{MAX}} \right\}, \max_j \left\{ S^{(j)}_{\mathrm{MIN}} \right\} \right\} \cdot e^{-\min_j d^{(j)}}\,. && (3.60)
\end{aligned}
$$

Variance index (VAR) [min.]

The Variance index [35] measures the compactness across clusters in the partition by averaging distances between cluster's members and its centroid

$$\mathrm{VAR}(\mathbf{C}) = \sqrt{\frac{1}{N} \sum_{\mathbf{c}_k \in \mathbf{C}} \sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k})}\,. \tag{3.61}$$

Xie-Beni index (XB) [min.]

Xie and Beni devised an index [60] that measures the partition compactness as average distance from each cluster member to the cluster's centroid and the separation as the minimum distance between clusters' centroids, as follows

$$\mathrm{XB}(\mathbf{C}) = \frac{\sum_{\mathbf{c}_k \in \mathbf{C}} \sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k})^2}{N \cdot \min_{\mathbf{c}_i, \mathbf{c}_j \in \mathbf{C}, j \neq i} d_E(\overline{\mathbf{c}_i}, \overline{\mathbf{c}_j})^2}\,. \tag{3.62}$$

Xie-Beni index - modified (XBmod) [min.]

Kim and Ramakrishna [54] proposed a modification of the Xie-Beni index, where the average cluster compactness, i.e. the numerator in Eq. (3.62), is substituted by the maximum cluster compactness resulting in the following equation

$$\mathrm{XBmod}(\mathbf{C}) = \frac{\max_{\mathbf{c}_k \in \mathbf{C}} \left\{ \frac{\sum_{\mathbf{x}_i \in \mathbf{c}_k} d_E(\mathbf{x}_i, \overline{\mathbf{c}_k})^2}{n_k} \right\}}{\min_{\mathbf{c}_i, \mathbf{c}_j \in \mathbf{C}, j \neq i} d_E(\overline{\mathbf{c}_i}, \overline{\mathbf{c}_j})^2}\,. \tag{3.63}$$

3.6 Experimental comparison of indices

In this section we experimentally compare our proposed DNs index with 33 other indices, described in Section 3.5. To evaluate the performance of each index, we employ four complementary methodologies on numerous synthetic and real-world datasets. Let us first describe them in more detail and then proceed to the protocol of the evaluation. Results will conclude this section.

3.6.1 Datasets

In our study we collected numerous datasets of different types and from various sources: synthetic datasets created using the `Complex2D` data generator described in Chapter 2, and real-world datasets. The latter are of two types: gene expression datasets and other, non-genetic ones. These three collections are used throughout the entire thesis to evaluate the cluster validation indices, clustering algorithms in Chapter 4, and ensemble approaches in Chapter 5.

Synthetic datasets

We used the whole `Complex2D` family, which is presented in Section 2.3 and consists of three sub-groups related to three different scenarios:

- `S1-Complex2D` (540 datasets),
- `S2-Complex2D` (540 datasets),
- `S3-Complex2D` (450 datasets).

So, we have 1530 synthetic datasets in total, each containing $N = 500$ data points in a two-dimensional space. The `Complex2D` datasets are scaled proportionally to fit into the square of a unit side length. For other details, see the description of the synthetic data generator described in Chapter 2.

Gene expression datasets

The second collection of datasets comes from the field of genomics, where the scientists employ micro-array technologies to measure the level of gene expression under different treatments. Cluster analysis proved to be very helpful in discovering new subtypes of diseases, especially cancer [86]. For the purpose of cluster validity indices comparison,

Table 3.1

Gene expression datasets used in the experiments, containing N samples in D dimensions. D equals the number of genes. The number of clusters K_T is a man-given ground-truth.

dataset name	N	D	K_T	reference
bladderCarcinoma	40	1203	3	[61]
breastCancer	49	1198	2	[62]
breastColonTumors	104	182	2	[63]
carcinomas	174	1571	10	[64]
centralNervousSystem	34	857	2	[65]
endometrialCancer	42	1771	4	[66]
glioblastomaMultiforme	37	1411	3	[67]
gliomagenesis	50	1739	3	[68]
gliomas	50	1377	4	[69]
hepatocellularCarcinoma	179	85	2	[70]
leukemia-1	248	2526	2	[71]
leukemia-2	72	1081	2	[72]
leukemia-3	72	1868	3	[73]
lungTumor-1	203	1543	5	[74]
lungTumor-2	66	4553	4	[75]
lymphoma-1	42	1095	2	[76]
lymphoma-2	77	798	2	[77]
melanoma	38	2201	2	[78]
mesothelioma	181	1626	2	[79]
multitissue	190	1363	14	[80]
prostateCancer-1	92	1288	4	[81]
prostateCancer-2	69	1625	3	[82]
prostateCancer-3	102	339	2	[83]
roundBluecellTumor	83	1069	4	[84]
serratedCarcinomas	37	2202	2	[85]

we selected 25 gene-expression datasets[6] that were previously preprocessed by de Souto

[6]The datasets were downloaded from `http://bioinformatics.rutgers.edu/Static/Supplements/CompCancer/datasets.htm`.

et al. [86]. They removed uninformative genes using feature-selection methods, thus reducing the dimensionality of the data substantially. The datasets are listed in Tab. 3.1 along with the information about the number of samples N, the number of genes or dimensions D, and the correct or expected number of clusters K_T as defined by the experts. The datasets are standardized by the z-score transformation, so that gene expression values, i.e. data features, have zero mean and unit variance. Duplicate data points are removed in a pre-processing step. We refer to this collection as `GENE`.

Real non-genetic datasets

Table 3.2

Real non-genetic datasets used in the experiments, containing N samples in D dimensions. The number of clusters K_T is a man-given ground-truth.

dataset name	N	D	K_T	reference
`breastCancerWisconsin`	683	9	2	[87]
`contractions`	98	27	2	[88]
`fertility`	100	9	2	[89]
`glass`	214	9	6	[90]
`ionosphere`	351	33	2	[91]
`iris`	150	4	3	[92]
`laryngeal3`	353	16	3	[88]
`letterRecognitionABC`	2291	16	3	[93]
`respiratory`	85	17	2	[88]
`segmentationTrain`	210	18	7	[94]
`voice_3`	238	10	3	[88]
`weaning`	302	17	2	[88]
`wine`	178	13	3	[95]
`wineQualityRed3`	1518	11	3	[96]
`yeast`	1484	8	10	[97]

Beside the `GENE` datasets, we gathered additional 15 real-world datasets that are non-genetic by their origin. For brevity, we call them `REAL` datasets and list them in Tab. 3.2. Five of them, namely `contractions`, `laryngeal3`, `respiratory`, `wean-`

ing, and voice_3 are obtained from the prof. Kuncheva research group[7] [88]. The other 10 datasets were downloaded from the UC Irvine Machine Learning Repository [94]. The breastCancerWisconsin dataset originally contains 699 samples, but 16 of them have missing values, so we removed them. The letterRecognitionABC dataset refers to the Letter Recognition dataset, where only 3 classes are retained: for letters A, B, and C. The segmentationTrain data are taken as a subset of the Image Segmentation dataset, where we only consider the training samples. We removed duplicate data points and standardized the datasets using the z-score transformation.

3.6.2 Evaluation protocol

A debate on how we should evaluate and compare CVIs is as old as indices themselves. In the literature, we identified three methodologies, to which we refer as *classical*, *alternative-all*, and *alternative-best*. The classical one [98] has been the default choice for decades [36, 39, 40, 46, 54, 99] and quite recently the alternatives were proposed by Vendramin et al. in 2009 [100] and Gurrutxaga et al. in 2011 [37].

Following the *classical* methodology, an index makes a successful guess about the best partition considering the set $\mathbf{P}$ of M partitions on a given dataset $\mathbf{X}$, if this partition contains the *true* number of clusters. The true number of clusters has to be known in advance, given as the ground-truth by the creator of the data or an expert. The best partition $\mathbf{C}^*$ predicted by the CVI is the partition that optimizes the index's value. Let us define $\mathbf{v}_I$ as a vector containing the values of CVI of each partition in the set $\mathbf{P}$, so

$$\mathbf{v}_I = \{\mathrm{CVI}(\mathbf{C}_1), \mathrm{CVI}(\mathbf{C}_2), \dots, \mathrm{CVI}(\mathbf{C}_M)\} \ . \tag{3.64}$$

In the case of max-like CVI, the best partition is obtained as follows

$$\mathbf{C}^* = \arg\max_{\mathbf{C}\in\mathbf{P}} \mathbf{v}_I \ , \tag{3.65}$$

and in the case of a min-like index

$$\mathbf{C}^* = \arg\min_{\mathbf{C}\in\mathbf{P}} \mathbf{v}_I \ . \tag{3.66}$$

Let K^* be the number of clusters in the partition $\mathbf{C}^*$, so $K^* = |\mathbf{C}^*|$, and let K_T denote the true number of clusters on a given dataset. It is said that the index has

[7] The datasets are available at http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm.

made a correct guess, if $K^* = K_T$. Let us define the score by the *classical* methodology as

$$S_{\mathrm{T}} = \begin{cases} 1, & \text{if } K^* = K_T \\ 0, & \text{otherwise} \end{cases} . \qquad (3.67)$$

As said before, two groups of researchers criticize such methodology and propose their alternatives: *alternative-all* [38, 49, 100] and *alternative-best* [11, 37]. They argue that classical methodology makes an important and sometimes false assumption that the clustering algorithm works well. In other words, it is expected that the partition that contains the true number of clusters fits the data better than any other partition in $\mathbf{P}$. This is often not the case, especially it is not true for the datasets with noise, complex-shaped or overlapped clusters. Both the alternatives share the idea of how to overcome these limitations: we should first ask an external validity index (eCVI) to provide objective scores of the partitions and then check to which degree the values of an internal index comply with them. However, the implementations of this idea vary. The *alternative-best* suggests we should consider only the partition, to which an internal index points as the optimal and determine, whether the eCVI agrees or not this is the best partition in the set. The *alternative-all* methodology considers all the partitions in the set – it computes the correlation between the values of an internal and external index and use this as a measure of index's quality. The more the CVI complies with the eCVI, the better the quality of the CVI. For the both alternative methodologies we need to know the *true* partition $\mathbf{C}^T$ of data that acts as a reference for external index.

Let us briefly formalize the alternative methodologies. Let $\mathrm{eCVI}(\mathbf{C}_i, \mathbf{C}_j)$ be an external validity index, called also a similarity measure, that returns high value, if the partitions $\mathbf{C}_i$ and $\mathbf{C}_j$ are similar and small value, if they are not similar. Let $\mathbf{v}_E$ be a vector of the eCVI values on a set of partitions $\mathbf{P}$

$$\mathbf{v}_E = \{\mathrm{eCVI}(\mathbf{C}_1, \mathbf{C}^T), \mathrm{eCVI}(\mathbf{C}_2, \mathbf{C}^T), \dots, \mathrm{eCVI}(\mathbf{C}_M, \mathbf{C}^T)\} . \qquad (3.68)$$

According to the *alternative-best* procedure [37], the most similar partition $\hat{\mathbf{C}}$ with respect to the true partition $\mathbf{C}^T$ is defined as

$$\hat{\mathbf{C}} = \arg\max_{\mathbf{C} \in \mathbf{P}} \mathbf{v}_E . \qquad (3.69)$$

We search among all the partitions made by a particular clustering algorithm to find the one, which is the most similar to the ground truth partition $\mathbf{C}^T$. It follows that the internal index CVI makes a successful guess, if it predicts $\hat{\mathbf{C}}$ as the best partition. So the score of the CVI is defined as

$$S_{\mathrm{BEST}} = \begin{cases} 1, & \text{if } \mathbf{C}^* = \hat{\mathbf{C}} \\ 0, & \text{otherwise} \end{cases} . \tag{3.70}$$

To compute the score of the internal validity index by means of *alternative-all* methodology, we consider all the values of CVI and eCVI, i.e. vectors $\mathbf{v}_I$ and $\mathbf{v}_E$, and compute correlation between them

$$S_{\mathrm{ALL}} = \mathrm{corr}(\mathbf{v}_I, \mathbf{v}_E) . \tag{3.71}$$

As a correlation function corr we employ the Kendall rank correlation coefficient τ [101]. Before calculating the correlation, we transform min-like indices into max-like by mirroring their values over their mean.

Moreover, we devise yet another criterion to assess the behaviour of a CVI - the score of *explicitness* S_{EXPL}. It measures how unambiguous or explicit an index is about the best partition it selects. Basically, we count how many optimums an index reaches over the set of partitions and then we transform the measure to fit into the interval $[0, 1]$, where 0 means that all index's values are identical and 1 means there is only one optimum. The latter is preferred as it means an index can distinguish between partitions of different quality. We define the score of explicitness as

$$S_{\mathrm{EXPL}} = 1 - \frac{|\max_{\mathbf{C}\in\mathbf{P}} \mathbf{v}_I| - 1}{|\mathbf{P}| - 1} \tag{3.72}$$

in the case of a max-like index. Otherwise, we replace the operator of maximum with minimum.

To partition the datasets, two well-known clustering algorithms were employed: the k-means algorithm (KM) [102], and the self-tuning spectral clustering algorithm with local scaling (Sp) [103]. Certainly, we could include more clusterers, perhaps also the proposed gSOM algorithm, but we do not focus here on the clustering process, rather on its validation. So, we have chosen these two methods due to the small number of parameters and their wide recognition by the community. Also, the Sp algorithm proved to perform well also when the clusters are of arbitrary shapes [103]. However, we performed exhaustive experiments on the gSOM algorithm already in Chapter 4 as well as in the following Chapter 5 on the same datasets. The KM algorithm is repeated 10 times for each experiment configuration due to its stochastic nature. The parameter `kNN` of the Sp algorithm, which controls the local scaling was set on a value of 2. Both algorithms require the expected number of clusters K to be given as an input parameter. In our experiments K goes from 2 to $K_{\max}$, which is a function of N and K_T bounded by 25 or by $K_T + 5$ if K_T is larger than 25. To be explicit

$$K_{\max} = \max\left\{\min\left\{\left\lceil\sqrt{N}\right\rceil, 25\right\}, K_T + 5\right\} . \qquad (3.73)$$

The overview of our experimental protocol is as follows:

1. Fetch a dataset $\mathbf{X}$ from the collection `S1-Complex2D`, `S2-Complex2D`, `S3-Complex2D`, `GENE`, or `REAL`. Ground-truth partition $\mathbf{C}^T$ is known.

2. Employ a clustering algorithm (KM and Sp) to create a partition $\mathbf{C}$ of the dataset $\mathbf{X}$ for every $K = 2, 3, \dots, K_{max}$.

3. Validate each partition with internal CVIs and using external eCVIs with $\mathbf{C}^T$ provided. Repeat steps 1–3 for all datasets.

4. Compute S_{ALL}, S_{BEST}, S_{T}, and S_{EXPL} scores and their averages over datasets, clustering algorithms, and eCVIs.

5. Follow the standard procedure for algorithms comparison on multiple datasets using statistical tests [104–106]. Compute ranks of CVIs for every dataset using an average S_{ALL} over clusterers and eCVIs. Average ranks over the datasets and test a null hypothesis that ranks are equal with the Friedman's non-parametric test [107], which makes no assumptions about the normality of the scores. If the null hypothesis is rejected, make a pairwise comparison between CVIs using post hoc Shaffer's procedure [108] with 95% confidence level, i.e. $\alpha = 0.05$. The Shaffer's test is the most powerful procedure we could feasibly perform with 34 CVIs in comparison.

3.6.3 External cluster validity indices

External cluster validation measures the matching between the clustering solution and known optimal grouping, which has to be given as a ground-truth or gold standard for a particular dataset. In the experiments in this thesis, we use three eCVIs: Adjusted Rand Index (ARI) [109], Adjusted Mutual Information (AMI) [110], and Balanced Clustering Accuracy (BCA). To formalize them, let us first define some prerequisites.

Let us have two partitions $\mathbf{C}_i = \{\mathbf{c}_1^{(i)}, \mathbf{c}_2^{(i)}, \dots, \mathbf{c}_U^{(i)}\}$ and $\mathbf{C}_j = \{\mathbf{c}_1^{(j)}, \mathbf{c}_2^{(j)}, \dots, \mathbf{c}_V^{(j)}\}$ of the same dataset $\mathbf{X}$. With U and V we denote the number of clusters in the partitions $\mathbf{C}_i$ and $\mathbf{C}_j$, respectively, so $U = |\mathbf{C}_i|$ and $V = |\mathbf{C}_j|$. We define $n_h^{(i)}$ as the number of data points in the cluster $\mathbf{c}_h^{(i)}$. Now, we count how many data points are clustered together in the same cluster in both partitions and write the numbers in a contingency table. Here, n_{hk} means the number of data points that are common for $\mathbf{c}_h^{(i)}$ and $\mathbf{c}_k^{(j)}$, i.e. $n_{hk} = |\mathbf{c}_h^{(i)} \cap \mathbf{c}_k^{(j)}|$.

$\mathbf{C}_i$ \ $\mathbf{C}_j$	$\mathbf{c}_1^{(j)}$	$\cdots$	$\mathbf{c}_k^{(j)}$	$\cdots$	$\mathbf{c}_V^{(j)}$	
$\mathbf{c}_1^{(i)}$	n_{11}	$\cdots$	n_{1k}	$\cdots$	n_{1V}	$n_1^{(i)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{c}_h^{(i)}$	n_{h1}	$\cdots$	n_{hk}	$\cdots$	n_{hV}	$n_h^{(i)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{c}_U^{(i)}$	n_{U1}	$\cdots$	n_{Uk}	$\cdots$	n_{UV}	$n_U^{(i)}$
	$n_1^{(j)}$	$\cdots$	$n_k^{(j)}$	$\cdots$	$n_V^{(j)}$	N

The ARI index is the Rand index (RI) [111] adjusted for chance. It means that if we fix the number of clusters U and V, and the number of data points in every cluster, and assign the cluster labels to the data points randomly, the ARI index should be zero under the assumption of generalized hypergeometric distribution. So, the ARI index measures how far we are from the assumption that the compared partitions $\mathbf{C}_i$ and $\mathbf{C}_j$ have occurred by chance. The adjustment for chance is achieved by subtracting the expected value E of the Rand index from itself and is defined as

$$\begin{aligned}\mathrm{ARI}(\mathbf{C}_i, \mathbf{C}_j) &= \frac{\mathrm{RI}(\mathbf{C}_i, \mathbf{C}_j) - E\left[\mathrm{RI}(\mathbf{C}_i, \mathbf{C}_j)\right]}{1 - E\left[\mathrm{RI}(\mathbf{C}_i, \mathbf{C}_j)\right]} = \\ &= \frac{\sum_{h=1}^{U}\sum_{k=1}^{V}\binom{n_{hk}}{2} - \left[\sum_{h=1}^{U}\binom{n_h^{(i)}}{2}\cdot\sum_{k=1}^{V}\binom{n_k^{(j)}}{2}\right]/\binom{N}{2}}{\frac{1}{2}\left(\sum_{h=1}^{U}\binom{n_h^{(i)}}{2} + \sum_{k=1}^{V}\binom{n_k^{(j)}}{2}\right) - \left[\sum_{h=1}^{U}\binom{n_h^{(i)}}{2}\cdot\sum_{k=1}^{V}\binom{n_k^{(j)}}{2}\right]/\binom{N}{2}} .\end{aligned} \tag{3.74}$$

The AMI index is based on information theory. It is derived from normalized mutual information index and is adjusted for chance like ARI. First, let us define the entropy H of a partition and the mutual information I between two partitions:

$$H(\mathbf{C}_i) = \sum_{h=1}^{U} \frac{n_h^{(i)}}{N} \cdot \log \frac{n_h^{(i)}}{N}\,, \tag{3.75}$$

$$I(\mathbf{C}_i, \mathbf{C}_j) = \sum_{h=1}^{U} \sum_{k=1}^{V} \frac{n_{hk}}{N} \cdot \log\left(\frac{N \cdot n_{hk}}{n_h^{(i)} \cdot n_k^{(j)}}\right). \tag{3.76}$$

Now, we have to compute the expected value E of the mutual information I between the partitions $\mathbf{C}_i$ and $\mathbf{C}_j$. For the detailed explanation and proof please see [112] – here we provide the following definition for completeness

$$\begin{aligned} E\left[I(\mathbf{C}_i, \mathbf{C}_j)\right] = \sum_{h=1}^{U} \sum_{k=1}^{V} \sum_{n_{hk}=\max\left\{n_h^{(i)}+n_k^{(j)}-N,0\right\}}^{\min\left\{n_h^{(i)},n_k^{(j)}\right\}} \frac{n_{hk}}{N} \cdot \log\left(\frac{N \cdot n_{hk}}{n_h^{(i)} \cdot n_k^{(j)}}\right) \cdot \\ \cdot \frac{n_h^{(i)}!\, n_k^{(j)}!\, \left(N - n_h^{(i)}\right)!\, \left(N - n_k^{(j)}\right)!}{N!\, n_{hk}!\, \left(n_h^{(i)} - n_{hk}\right)!\, \left(n_k^{(j)} - n_{hk}\right)!\, \left(N - n_h^{(i)} - n_k^{(j)} + n_{hk}\right)!}\,. \end{aligned} \tag{3.77}$$

The AMI index measures the mutual information between the two partitions and is normalized by the maximum entropy of those partitions. Moreover, to adjust the index for chance we subtract the expected mutual information from denominator and nominator of the ratio and we get the following equation

$$\mathrm{AMI}(\mathbf{C}_i, \mathbf{C}_j) = \frac{I(\mathbf{C}_i, \mathbf{C}_j) - E\left[I(\mathbf{C}_i, \mathbf{C}_j)\right]}{\max\left\{H(\mathbf{C}_i), H(\mathbf{C}_j)\right\} - E\left[I(\mathbf{C}_i, \mathbf{C}_j)\right]}\,. \tag{3.78}$$

In this thesis we also introduce a well-known performance measure for classifiers – the balanced accuracy [113] – in the context of clustering. Similar attempts were made before by Meilă, who adapted the classification accuracy to fit into the field of cluster analysis [114]. To emphasize the conceptual difference between classification and clustering we refer to the proposed measure as the Balanced Clustering Accuracy

(BCA). While the cluster labels are interchangeable, $\mathbf{c}_1^{(i)}$ from the partition $\mathbf{C}_i$ does not necessary correspond to the cluster $\mathbf{c}_1^{(j)} \in \mathbf{C}_j$. Therefore, we have to align clusters of both the partitions in comparison in the first place. This is known as the assignment problem and is solved iteratively by the Hungarian algorithm [115]. The result is a contingency table, where the diagonal elements are the number of common data points for two clusters. Note, that we have to determine, which of the partitions $\mathbf{C}_i$ and $\mathbf{C}_j$ is the reference one or ground-truth, as the BCA measure is not symmetric: $\mathrm{BCA}(\mathbf{C}_i, \mathbf{C}_j) \neq \mathrm{BCA}(\mathbf{C}_j, \mathbf{C}_i)$. Here, we assume that $\mathbf{C}_j$ is the reference partition. Finally, we define the balanced accuracy as the arithmetic mean of the cluster-wise accuracies:

$$\mathrm{BCA}(\mathbf{C}_i, \mathbf{C}_j) = \frac{1}{V} \sum_{k=1}^{V} \frac{n_{kk}}{n_k^{(j)}} \,. \tag{3.79}$$

All the eCVIs' values in this study are normalized on the interval $[0, 1]$. The value of 1 means that partitions $\mathbf{C}_i$ and $\mathbf{C}_j$ match perfectly and 0 that these two partitions completely disagree. ARI and AMI are adjusted for chance and the previous sentence is true in a stochastic sense: if the value of ARI or AMI is 0, it means that one of the comparing partitions $\mathbf{C}_i$ or $\mathbf{C}_j$ is supposed to equal a partition, where data points are randomly assigned to the clusters.

3.6.4 Results

For each dataset collection we produced two figures with a) the averaged scores S_{ALL}, S_{BEST}, S_{T}, and S_{EXPL} of each CVI in comparison b) along with its average rank based on the S_{ALL} as the most comprehensive of the scores. Furthermore, we observe whether the pair-wise differences between the CVIs are statistically significant or not using methodology for comparing multiple algorithms on multiple datasets [11, 105].

`Complex2D` *datasets*

Remember, we created three scenarios for synthetic data generator, where we graduate the complexity of the clustering problem with increasing number of linearly non-separable clusters L and the distance between each cluster pair d_{min}.

Let us start with the datasets of the first scenario `S1-Complex2D`. In Fig. 3.8 we show the averaged scores of each CVI in four separate bar charts. The values are sorted by the score S_{ALL} that is our strongest performance measure due to its ability to consider the whole set of partitions. We notice a discrepancy between the scores, for instance let us consider the top five indices SIL, DNs, SEP, CON, and I. In the case of SIL, DNs and I index, the S_{ALL} scores are in agreement with the S_{BEST} and S_{T} being relatively high compared to others. The contrary is true for SEP and CON – while having high correlation with eCVIs resulting in high S_{ALL}, they fail detecting the best partition or the *true* one resulting in low S_{BEST} and S_{T} scores, respectively. If we take a look on the S_{EXPL} score that measures the ambiguity of an index, we see that CON and HOMmin are the most ambiguous, but the differences are not so obvious in general. The performance of the proposed index DNs is comparable to the best CVIs regarding all the scores and is better than its relatives, i.e. the DN, DNg, and GDN indices.

Furthermore, we perform a test of the null hypothesis that CVIs do not differ in the performance using average ranks of the S_{ALL} scores. Friedman's test shows strong evidence for statistical significant differences for all the scenarios S_1, S_2, and S_3 with p-values below 10^{-300}, so we can safely reject the null hypothesis. In Fig. 3.9 we show a diagram of average ranks for the scenario S_1. The CVIs that do not perform significantly different are connected with a red bar. For instance, we cannot tell that there is a significant difference between the SIL and DNs index, but we can for SEP and SEPmax. We see there are three groups of CVIs for which is true that there is no overlapping of bars in between: the group of the best CVIs (SIL, DNs, CON, I, SEP), the worst CVIs (HOM, SF, VAR), and the largest group in the middle.

Now, let us examine the results from scenario S_2 and S_3. Fig. 3.10 and Fig. 3.11 correspond to datasets `S2-Complex2D` and the results of the `S3-Complex2D` datasets are depicted in Fig. 3.12 and Fig. 3.13. Indeed, the reader is kindly encouraged to investigate the graphs in more detail with focus on his or her preferred CVI – however, we try to draw some global conclusions concentrating on CVIs on the top and the bottom of the list.

The first observation is that the list of the CVIs regarding the S_{ALL} score is changing consistently when we switch over the scenarios, meaning that some indices are constantly gaining places in the list and the others are moving down. For example, DNs, DNg, and DN are placed higher in the scenario S_3 then in the S_2 or S_1. Also, the GDN indices prosper under S_2 and S_3 scenarios. On the other hand, the winner of the S_1, the SIL index, degrades in performance when the complexity of the datasets increases. The same is true for DB and DBmod.

Secondly, the pairwise statistical test shows some more significant differences between CVIs, especially on the top of the rank list, where two smaller groups have formed. Considering S_2 scenario, DNs, SEP, and CON are in the best performing group, followed by the SEPmax and I index. The DNs index is significantly better than all the other CVIs, except for the SEP index. However, in the scenario S_3, the proposed index appears to be a clear winner. In the group of the least successful indices (HOM, SF, VAR), there are no noticeable differences.

Finally, we observe a rather big drop in overall performance considering S_{BEST} and S_{T} scores when moving from scenario S_1 to S_2 and S_3. Indices DNs, DN, I and some members of the GDN family prove to be better than others in this scope of analysis.

GENE *datasets*

Gene expression datasets are very different from two-dimensional synthetic data we discussed in the previous section for they have many more dimensions than there are data samples. This property of the GENE datasets poses a great challenge to cluster analysis as can be seen also from the results in Fig. 3.14. The correlation between CVIs values and the external evaluation of eCVIs measured by the S_{ALL} score is quite low for all the CVIs. Nevertheless, S_{BEST} and S_{T} are not so pessimistic and have greater variation across CVIs. Here, the S_{EXPL} points out an outlier SF, which tends to report many partitions as optimal and is therefore marked as the most ambiguous. The proposed index DNs proves to be among the best ones and is better than its closest relatives DNg and DN.

The results of a pairwise comparison using average ranks of S_{ALL} score are depicted in Fig. 3.15, showing there are no significant differences among CVIs. Indeed, Friedman's omnibus statistical test strongly rejects the null hypothesis of equal ranks with p-value less than $4 \cdot 10^{-5}$ showing at least one pair of CVIs varies significantly. But the subsequent Shaffer's post hoc test at 5% confidence interval cannot find specifically which pair is that. If we loosen the confidence interval to 10%, Shaffer's test discovers significant difference between the last GDN[4,1] and the pair of the best performing CON and GDN[2,2]. Other differences in average ranks are not significant. However, we can still say something about the ranking of the CVIs. Like in the case of Complex2D datasets, the indices CON, DNs, I and SEPmax are being preferable along with some newcomers at the top, namely CH and some GDN indices.

`REAL` *datasets*

The results of the CVIs' performance on another real-world datasets are gathered in Fig. 3.16 and Fig. 3.17. Here, the CH index is quite a surprise for not being so successful on `Complex2D` datasets. Considering the average scores of all the methodologies, i.e. S_{ALL}, S_{BEST}, and S_{T}, we observe that I, SEP, CON, DNs, and SEPmax are among the best indices and VAR, SDBW, HOM, and CI performs poorly. DNs proves to be better than DN, DNg, and GDN indices regarding S_{ALL} and is better than DN and DNg also when we consider S_{BEST} or S_{T}. As can be seen from Fig. 3.17, DNs performs significantly better than its ancestor DN as well as other eleven CVIs.

3.7 Conclusion

In this chapter we motivated the usage of a cluster validation step as essential in cluster analysis. We have proposed an evolved or modified version of the Dunn's validity index DNs that is based on the shortest paths on the Gabriel graph. It features a penalization for large number of clusters, which are in practice undesirable. To evaluate the strength of DNs in synthetic and real-world use-cases, we conducted a comparison of 34 cluster validity indices using three complementary performance measures from the existing body of literature along with a *score of explicitness* S_{EXPL}. As there are more than one possible way of evaluating the CVIs, we advocate to use them all but to give the emphasis on the most comprehensive one, which is in our opinion the *alternative-all* methodology, i.e. the S_{ALL} score.

Our study is one of the largest in the literature [11, 37, 38, 98], taking into account the number of indices, datasets and performance measures. To the best of our knowledge it is the first study that systematically addresses the performance of CVIs in a case of datasets with linearly non-separable clusters. To assess such scenarios we employed our novel algorithm for data generation described in Chapter 2 and showed the differences in behaviour of compared CVIs when the degree of clusters' entanglement increases.

From the experimental results we can conclude that the proposed DNs index always outperforms its ancestors, the DN and DNg indices. The improvement in performance is statistically significant for the majority of the cases. Based on the experiments on `Complex2D`, `GENE` and `REAL` datasets we also identified an approximate set of preferable indices that show good performance consistently: CON, SEP, SEPmax, I, and DNs.

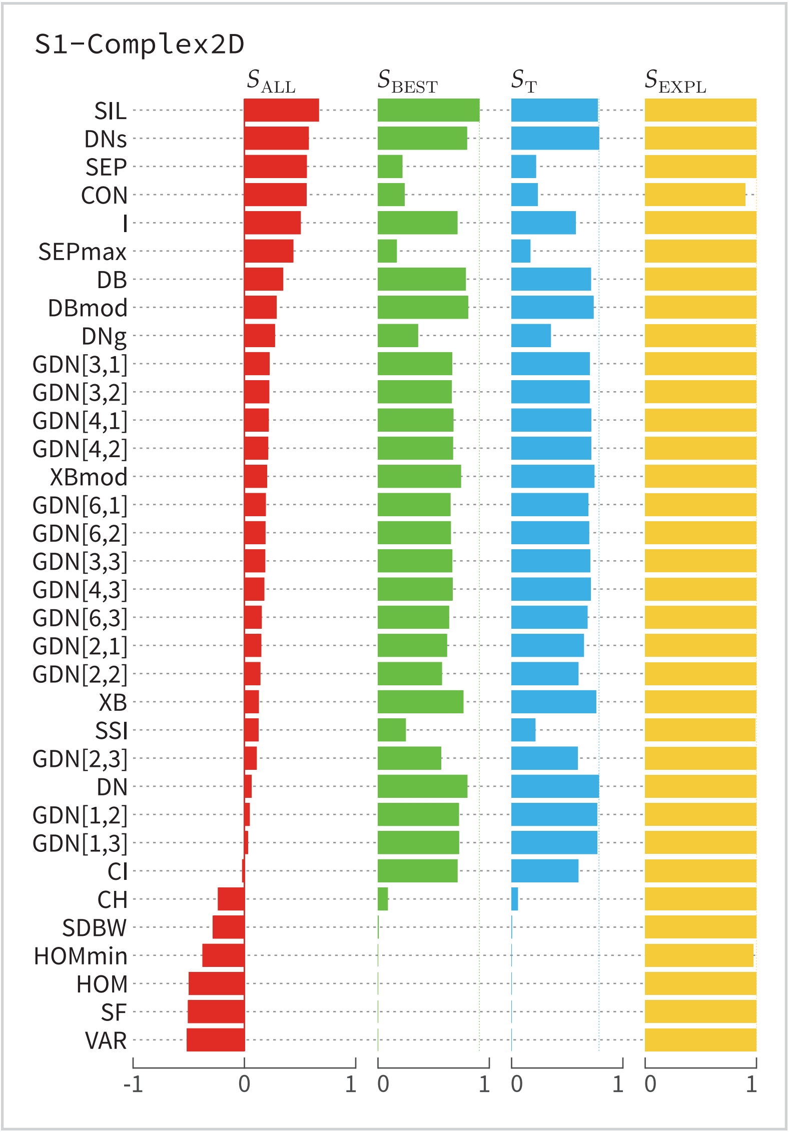


Figure 3.8

Overall scores averages for datasets from `Complex2D` collection, scenario S_1. Indices are sorted by the value of S_{ALL}.

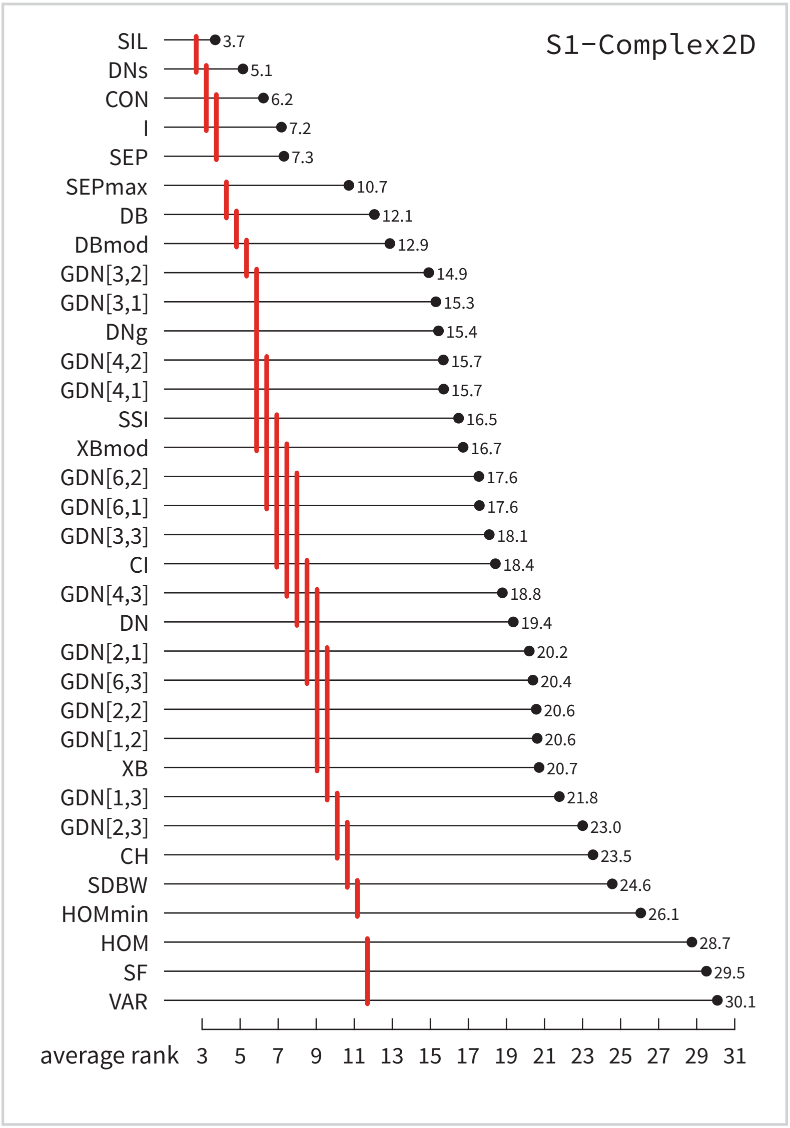


Figure 3.9

Average ranks of validity indices based on the average of Kendall's correlation (S_{ALL}) across all the datasets from the `Complex2D` collection, scenario S_1. Red lines connect indices with no significant difference in performance. We used Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$.

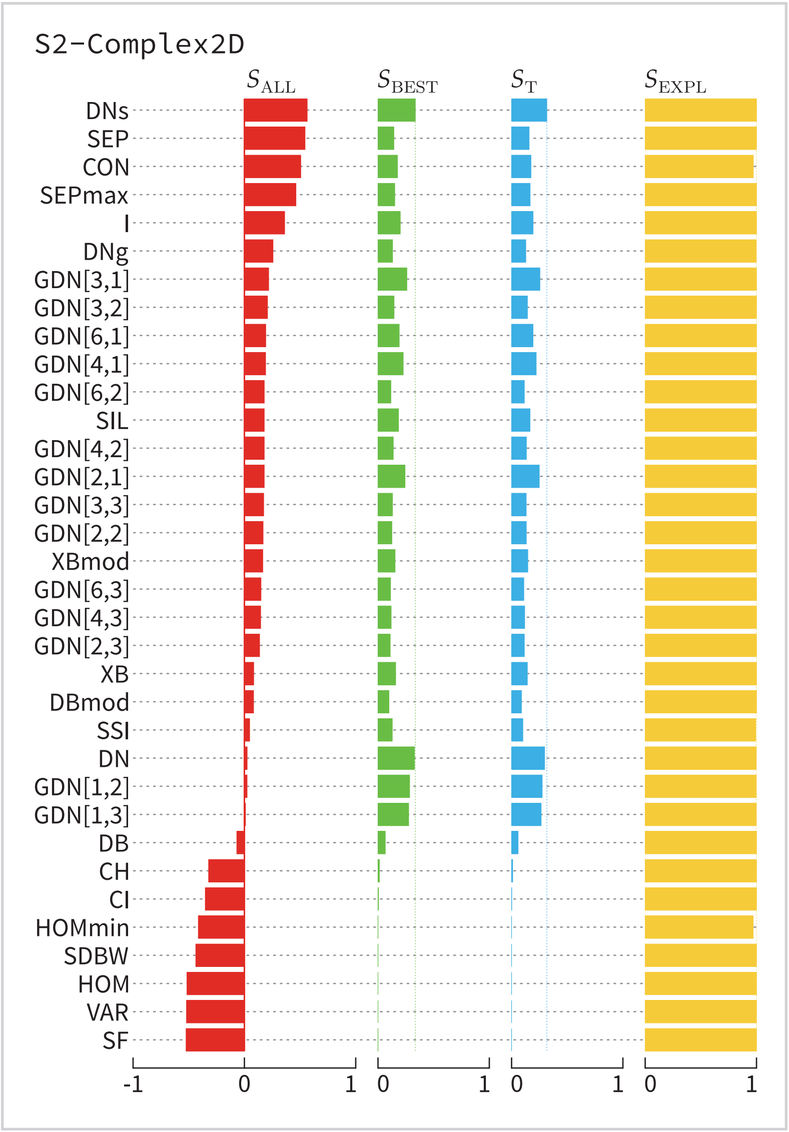


Figure 3.10

Overall scores averages for datasets from `Complex2D` collection, scenario S_2. Indices are sorted by the value of S_{ALL}.

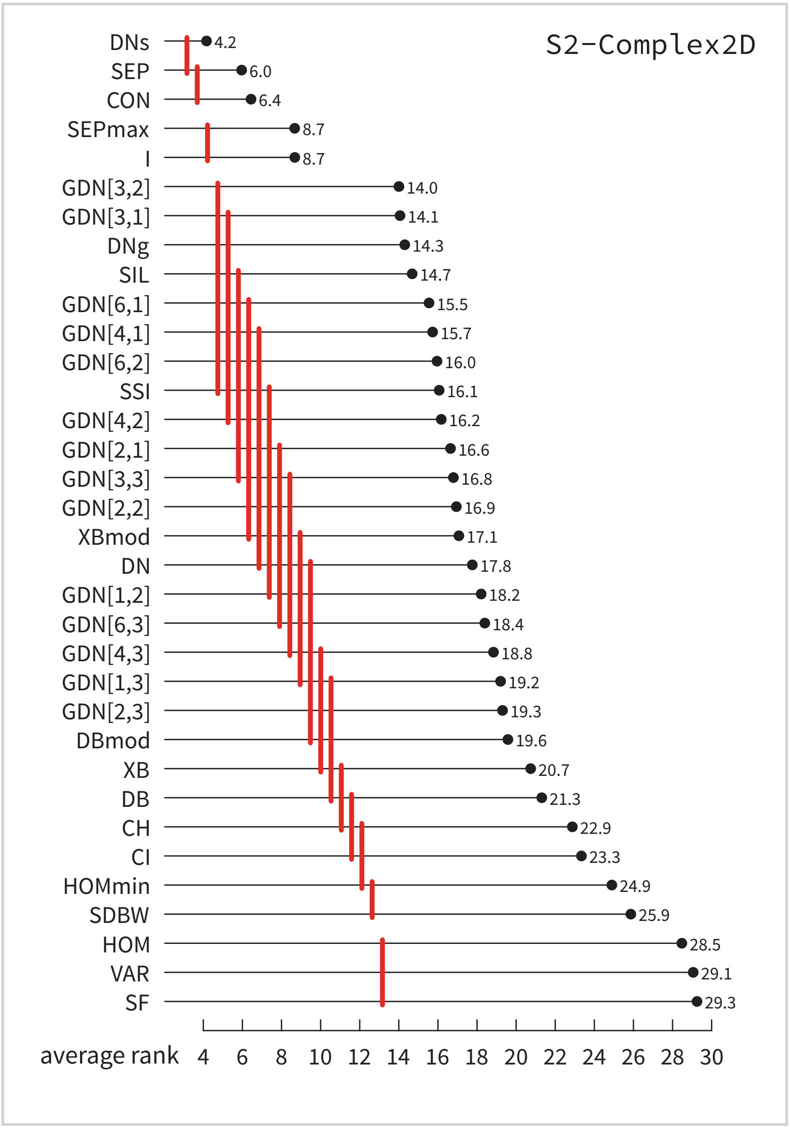


Figure 3.11

Average ranks of validity indices based on the average of Kendall's correlation (S_{ALL}) across all the datasets from the `Complex2D` collection, scenario S_2. Red lines connect indices with no significant difference in performance. We used Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$.

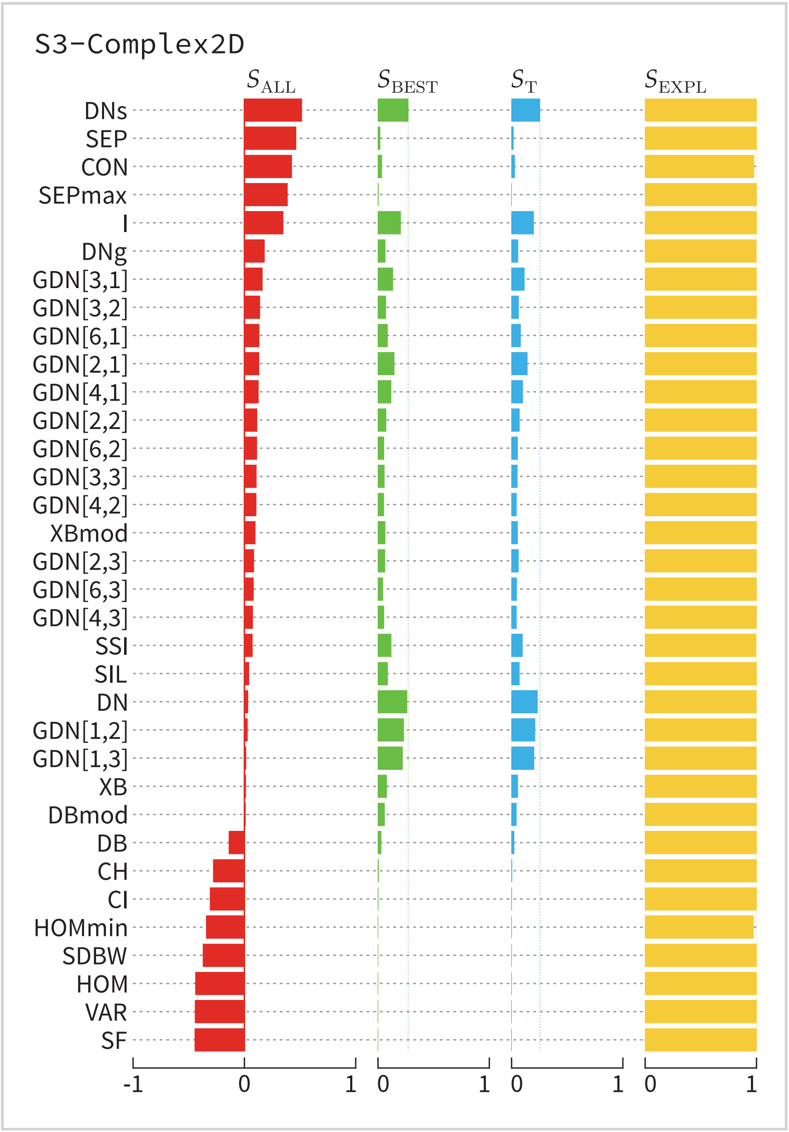


Figure 3.12

Overall scores averages for datasets from `Complex2D` collection, scenario S_3. Indices are sorted by the value of S_{ALL}.

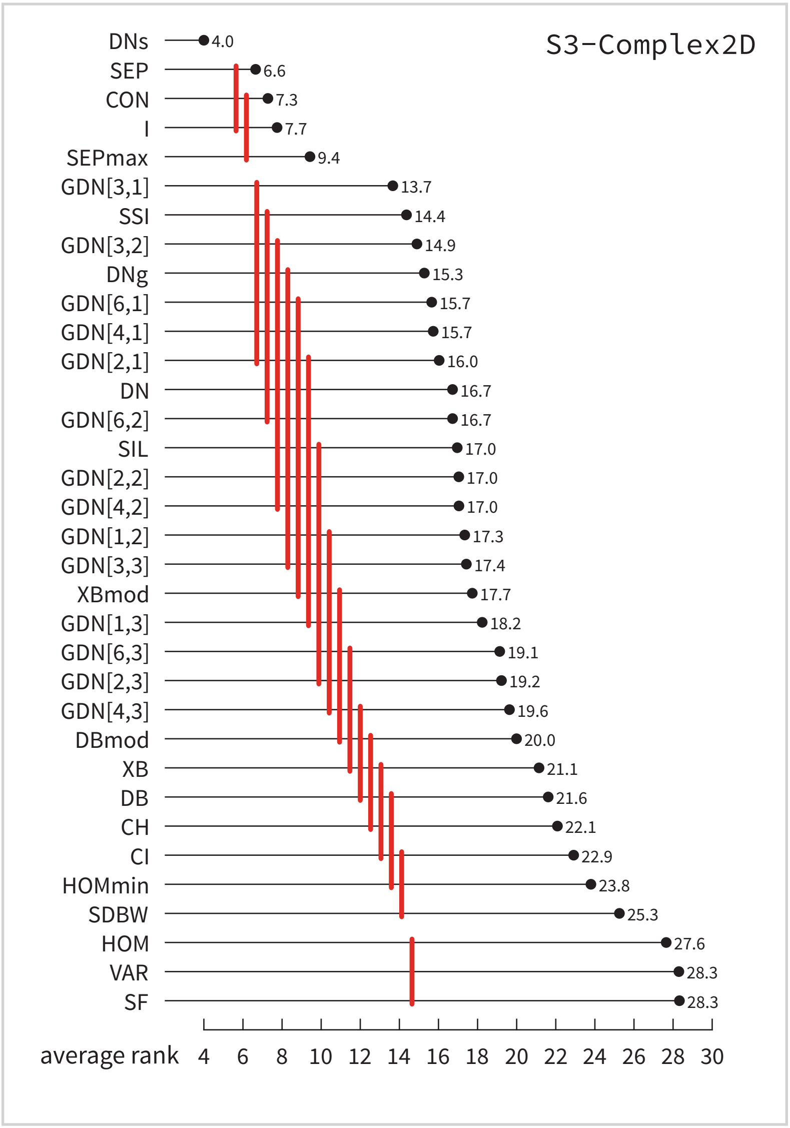


Figure 3.13

Average ranks of validity indices based on the average of Kendall's correlation (S_{ALL}) across all the datasets from the `Complex2D` collection, scenario S_3. Red lines connect indices with no significant difference in performance. We used Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$.

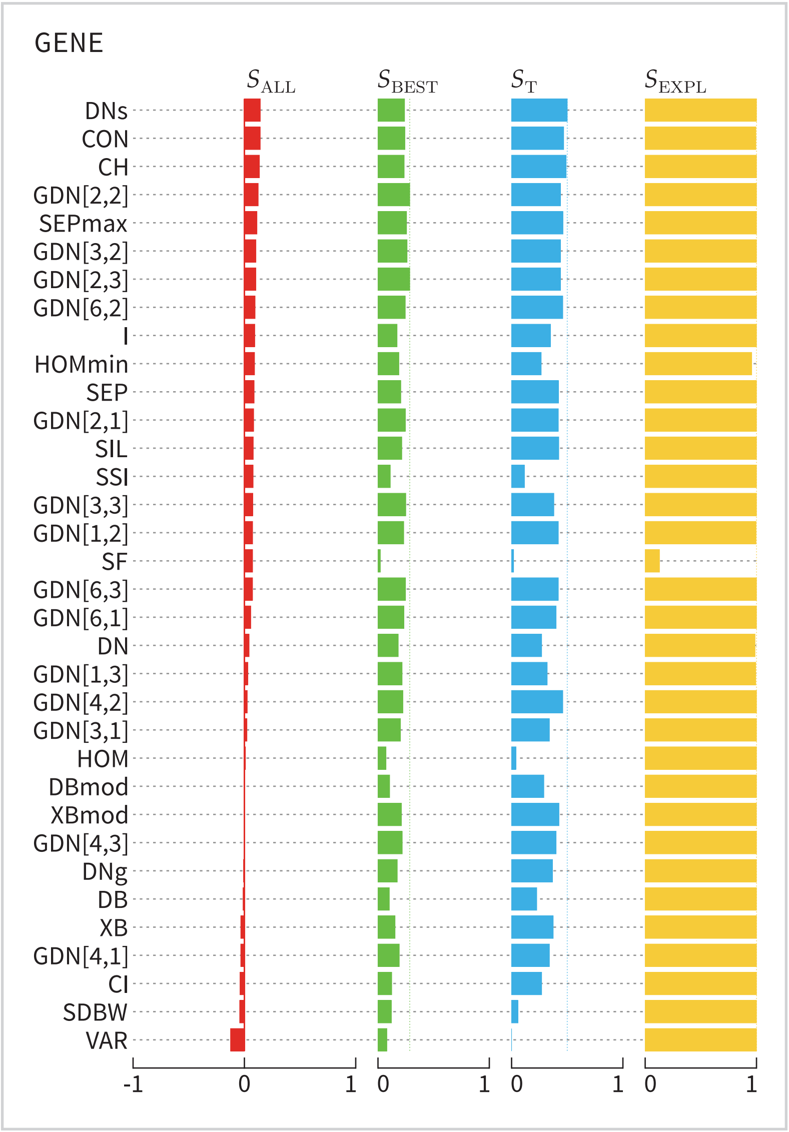


Figure 3.14

Overall scores averages of GENE datasets. Indices are sorted by the value of S_{ALL}.

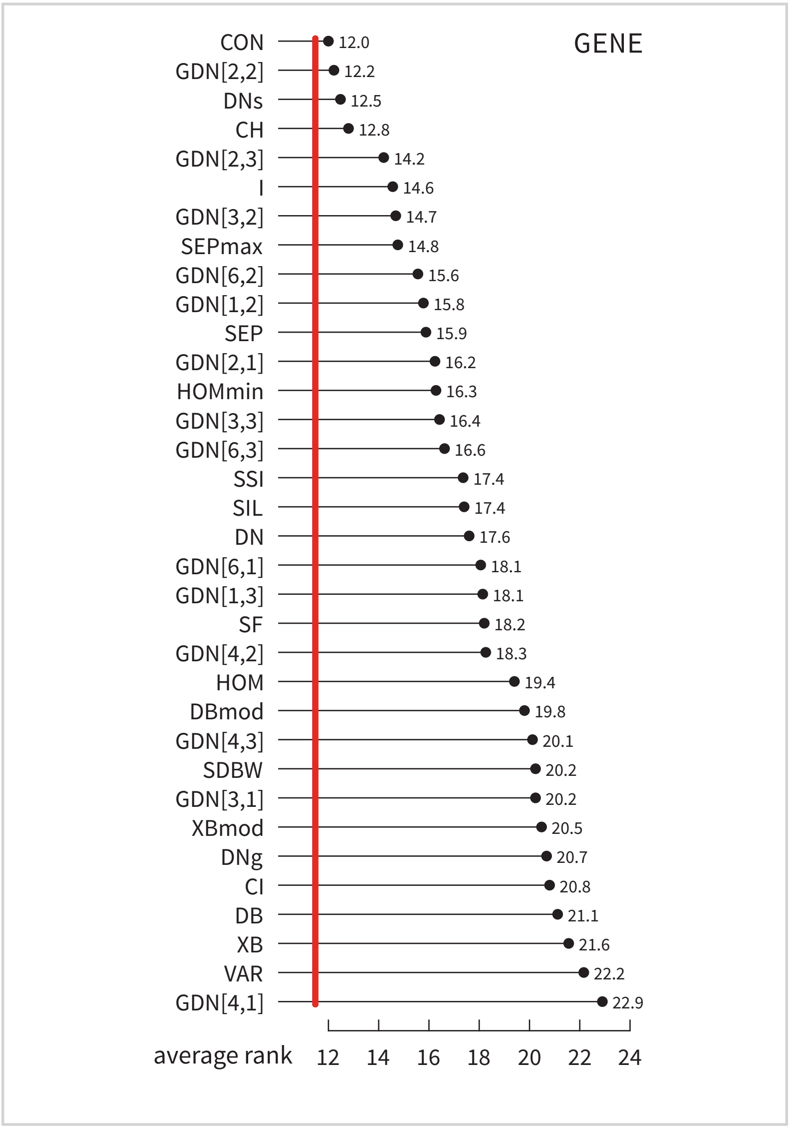


Figure 3.15

Average ranks of validity indices based on the average of Kendall's correlation (S_{ALL}) across all the datasets from the GENE datasets. Red lines connect indices with no significant difference in performance. We used Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$.

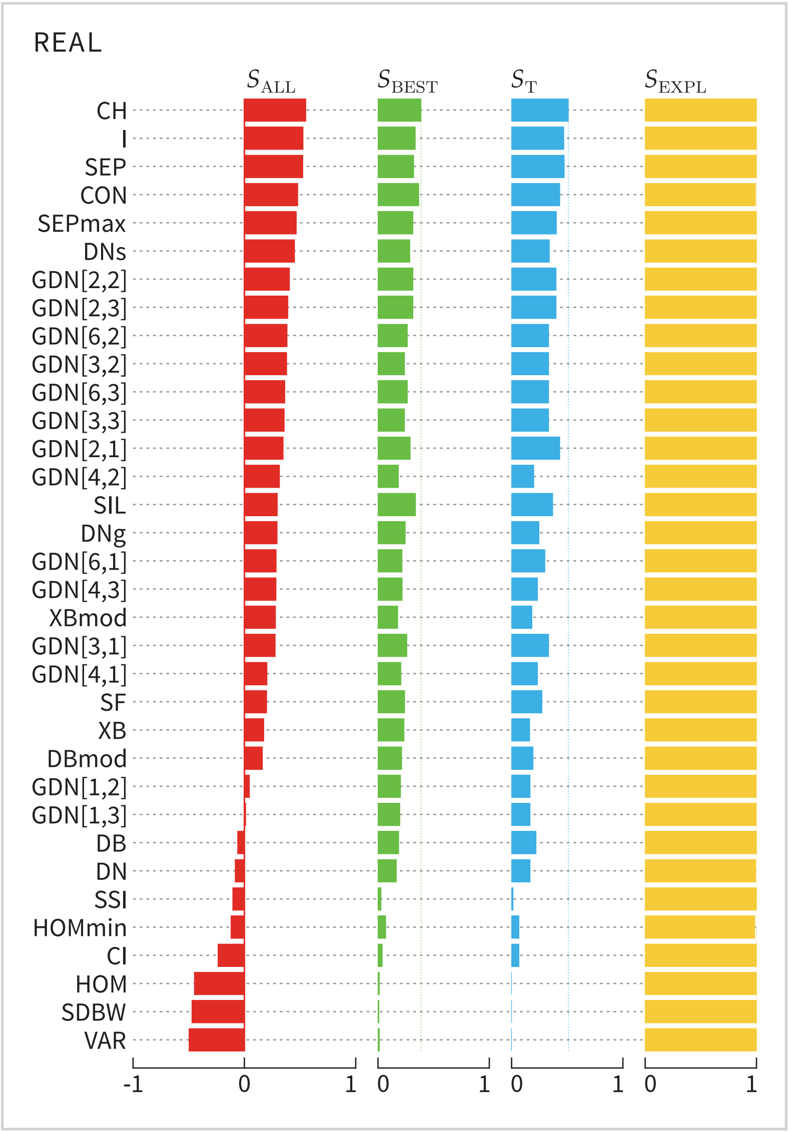


Figure 3.16

Overall scores averages of REAL datasets. Indices are sorted by the value of S_{ALL}.

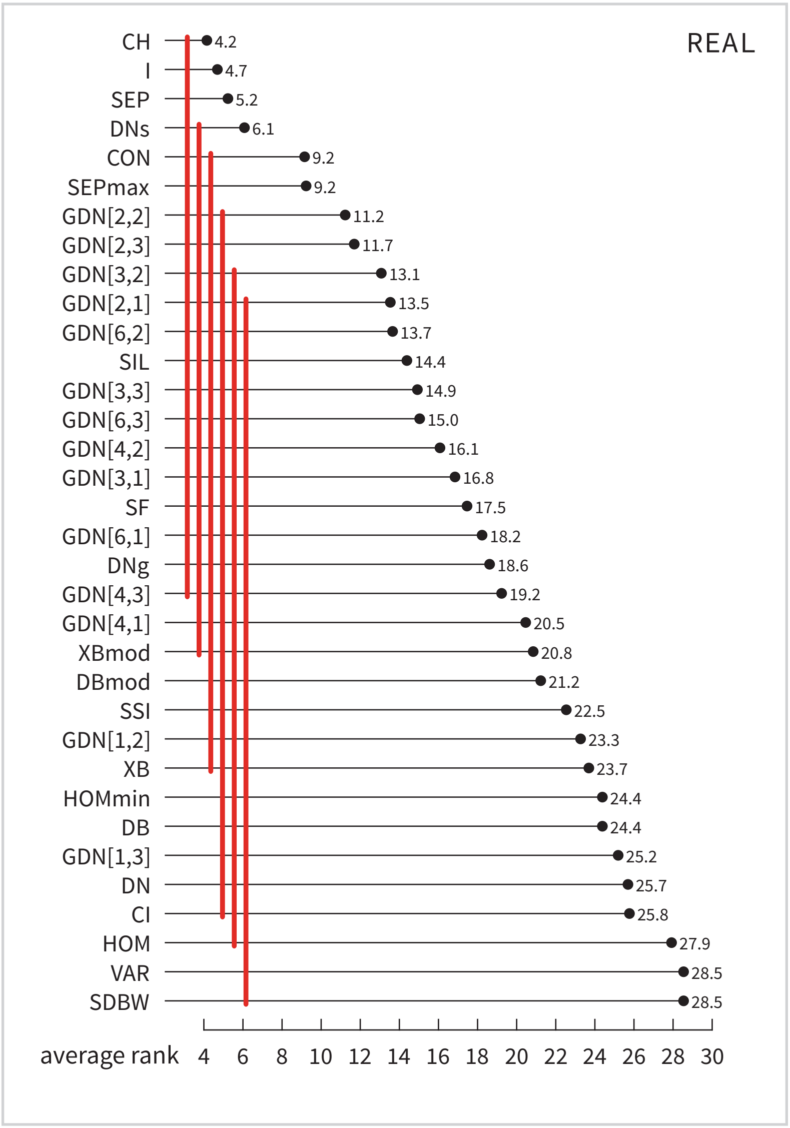


Figure 3.17

Average ranks of validity indices based on the average of Kendall's correlation (S_{ALL}) across all the datasets from the REAL datasets. Red lines connect indices with no significant difference in performance. We used Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$.

4

Gravitational Clustering of Self-organizing map

4.1 Introduction

In this chapter we describe a novel tool for data clustering – the gSOM[1] algorithm – and show its abilities on data from various problem domains. The proposed algorithm is based on clustering of the self-organizing map (SOM) proposed by Kohonen [16], which is one of the most popular neural-network approaches for data clustering [116]. The SOM is primarily an efficient data visualization tool that projects the input data into two or three-dimensional space. Nevertheless, SOM can also be used as a prominent clustering method, where a trained map of neurons provides a data-abstraction layer that serves as a basis for the identification of clusters in the underlying data [117]. In order to obtain the high-level cluster structure of the input data, we have to find a grouping of neurons that are representatives of these data points.

Here, we present a new approach to the clustering of the SOM that relies on a nature-inspired gravitational algorithm that simulates the simplified Newtonian law of gravity in order to cluster objects [118]. Our objective is to fully exploit the advantages, both of the SOM as a data-abstraction method and the gravitational approach, in order to devise an efficient clustering algorithm capable of finding clusters of arbitrary shape and determining their number automatically. The latter is especially useful when dealing with unknown data or when the user does not have adequate experiences to correctly interpret the SOM results.

Let us start with some basic preliminaries about the self-organizing map and then continue with a review of the clustering methods that are based on SOM. After that, we introduce our attempt to integrate the gravitational algorithm in a two-level scheme of clustering of SOM. Afterwards, we compare the proposed gSOM algorithm theoretically and experimentally with seven related algorithms on synthetic and real-world datasets.

4.2 Self-organizing map

The self-organizing map is an unsupervised artificial neural-network model that implements the topology-preserving non-linear mapping of the high-dimensional input space to a regular low-dimensional grid of neurons with a defined neighbourhood [119, 120]. As such, it serves as an idealized model of certain neural structures in

[1]The acronym gSOM stands for gravitational clustering of Self-Organizing Map and should not be confused with the Growing Self-Organizing Map, usually abbreviated as GSOM.

human brain [121]. SOM is a competitive-learning algorithm, meaning that the neurons compete with each other in the approximation of the input space. During learning phase, neurons adapt their weights according to the distribution of the input data samples. The weight vectors can be seen as data prototypes that fit on the input data like an elastic net. Since its invention, SOM has been applied in numerous fields of science: for the prediction of corporate bankruptcy in economics [122], for recommendation system of web services [123], in analysing urban lifestyle and residential choices [124], in the research of battery packs for electric vehicles [125], in digital forensics for criminals detection [126], in the environmental study of correlation between land use and water quality [127], in oceanography [128], and in drug discovery [129] to name just few recent studies. In the following, we describe SOM in more details and provide with the implementation particulars we used in the thesis.

SOM consists of $U = a \cdot b$ neurons or units, ordered in a regular two-dimensional grid $a \times b$. Each neuron is represented by its weight vector $\mathbf{w}_i = [w_{i1}, \dots, w_{iD}]$, where D is the dimension of the input data $\mathbf{X}$. Each neuron, except if it lies on the border of the map, is connected to four or six direct neighbours – it depends on choosing a rectangular or hexagonal grid structure, respectively.

To speed up the convergence of the SOM training, linear initialization of the map is made in the subspace spanned by the two eigenvectors with the greatest eigenvalues computed from the original data [16]. For the initialization and training of SOM, the SOM Toolbox for MATLAB[2] was used. We trained SOM in a batch mode [121], where the whole dataset is presented to SOM before the adjustments are made to the weight vectors. In each epoch, the dataset is partitioned according to the Voronoi regions of the neurons, meaning that each data point $\mathbf{x}_j \in \mathbf{X}$ belongs to the neuron to which it is the closest. This neuron is called the winning neuron, denoted with $c(j)$, so the following is true

$$d_E(\mathbf{x}_j, \mathbf{w}_{c(j)}) = \min_i d_E(\mathbf{x}_j - \mathbf{w}_i)\,, \tag{4.1}$$

where $d_E(\cdot, \cdot)$ is the Euclidean distance between two vectors. After the assignment of winning neurons, the weight vector of a neuron i, i.e. $\mathbf{w}_i$, is replaced by the mean value of all the data points in the neighbourhood defined by the Voronoi regions. The new

[2]SOM Toolbox 2.0 for MATLAB is available under the GNU General Public License at `http://www.cis.hut.fi/somtoolbox/`.

value of $\mathbf{w}_i$ is calculated as

$$\mathbf{w}_i(t+1) = \frac{\sum_{j=1}^{N} h_{i,\,c(j)}(t)\mathbf{x}_j}{\sum_{j=1}^{N} h_{i,\,c(j)}(t)}\,, \tag{4.2}$$

where N is the number of points in a dataset $\mathbf{X}$ and $h_{i,\,c(j)}(t)$ is the kernel function, used to describe the neuron's neighbourhood. The new value of the i-th weight vector $\mathbf{w_i}$ is computed using Eq. (4.2) as a weighted average of all data points, where each data point's weight equals to the value of the neighbourhood kernel function $h_{i,\,c(j)}(t)$ centred on the winning neuron $c(j)$ of this particular data point.

Note that the new value of the neuron's weight $\mathbf{w}_i(t+1)$ does not directly depend on the previous weight's value $\mathbf{w}_i(t)$. The latter has rather an implicit influence through the winning neurons' assignment in Eq. (4.1) and consequently, through the value of the neighbourhood kernel function $h_{i,\,c(j)}(t)$. We used a Gaussian kernel as a neighbourhood function with the width defined by the parameter σ that decreases monotonically with time

$$h_{i,\,c(j)}(t) = \exp\left(-\frac{d_E(\mathbf{r}_{c(j)}, \mathbf{r}_i)^2}{2\sigma^2(t)}\right)\,, \tag{4.3}$$

where $\mathbf{r}_{c(j)}$ and $\mathbf{r}_i$ are the positions of the neurons $c(j)$ and i on the SOM grid. The initial value for σ is set heuristically to $\sigma(t=0) = \sigma_0 = \max\{1, \lceil \max\{a, b\}/8 \rceil\}$. The number of neurons in each dimension of the map, a and b, is chosen in the following way

$$\frac{a}{b} \approx \sqrt{\frac{\lambda_1}{\lambda_2}}\,, \tag{4.4}$$

where λ_1 and λ_2 are the two largest eigenvalues from the covariance matrix of the input data [16]. SOM is trained in two phases: a rough phase with a number of epochs $l_r = \max\{1, \lceil 10 \cdot U/N \rceil\}$ and a fine-tuning phase, where the number of epochs is $l_f = \max\{1, \lceil 40 \cdot U/N \rceil\}$. Above, $U = a \cdot b = S \cdot \left\lceil 5\sqrt{N} \right\rceil$, where S is a scale factor set to 1 by default. The heuristics for choosing the values of σ_0, a, b, l_r, l_f and U are adopted from the authors of the SOM Toolbox [130].

Let us demonstrate the principles of SOM on an illustrative example that is shown in Fig. 4.1. We created a two-dimensional dataset with three clusters in the shape of letters L, C, and A, thus we call this a demonstrative dataset LCA. In the left panel in the figure, the data points are represented by grey dots. We applied the SOM algorithm using hexagonal grid and the scale factor $S = 2$ – thus, the size of the map becomes 15 x 10. The winning neurons are depicted as red dots and the connections between them by solid black lines. The neurons that are not winning for any data point are called interpolating neurons and are depicted as empty dots. The right panel shows only the winning neurons with connections between them. The reason, why we have removed interpolating neurons, we discuss in Section 4.4, where we describe the gSOM algorithm.

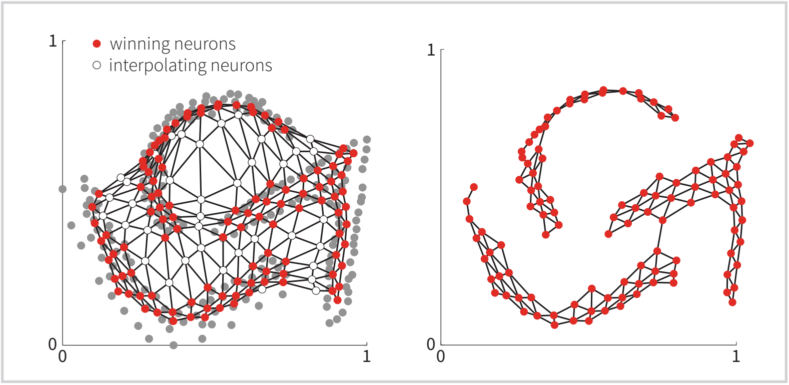


Figure 4.1

The trained SOM on the dataset LCA (left) and the winner neurons only with connections between them (right). The data points are drawn as grey dots, the red dots are winning neurons and the empty dots are interpolating neurons. The connections between neighbouring neurons in the SOM are depicted as black lines.

To sum up, we have obtained the trained SOM network of neurons. Each neuron covers a portion of data points, so it is a data representative and can be also seen as a proto-cluster. Our mission is now to group these proto-clusters into clusters. In the next section, we present some approaches for the clustering of the SOM neurons.

4.3 Two-level clustering of Self-organizing map

Since 1982, when SOM was proposed by Kohonen, many researchers have been using it for clustering purposes. In the early attempts, SOM was treated as an alternative to k-means and was applied to the data in a similar way: a one or two-dimensional map with K neurons was trained and each neuron on the output layer represented one cluster [131–134]. Pal et al. argued that feature mapping of SOM is conceptually different from clustering and one should not mix them together [135]. Moreover, Kiang found the attempts to cluster data with SOM problematic too, for it is difficult to design a two-dimensional map for datasets, where the number of clusters is a prime number [136]. Consequently, the machine-learning community made a noticeable shift towards the bigger feature maps and the idea of two levels in the clustering process: firstly, SOM is trained with few times more neurons than the expected number of clusters; and secondly, those neurons are clustered into the desired number of clusters. And, at last, the SOM and clustering methods "live" in a symbiosis. We can talk thereafter about the clustering of SOM or, equivalently, about the clustering with SOM – it depends on our point of view.

To the best of our knowledge, Ultsch and Simeon were the first to propose an idea of two-level cluster discovery in the data using SOM in 1990 [137]. They introduced a new visualization method of SOM with a unified distance matrix or U-matrix. The U-matrix consists of the relative distances between the neighbouring neurons in the input data space and in literature we can see three flavours of it. In the first one, there is one entry in the U-matrix for each neuron with the value of average distance to its neighbours [21, 138]. If we double the grid density, we get the second flavour of U-matrix representation. Here, some cells represent neurons and contain the average distances to the neighbours, as before. Other cells contain the precise distance between two neighbouring neurons [130, 137]. The third way is the same as the second one except that we do not display the average distances between neurons, so the entries in the U-matrix that correspond to neurons are empty [139]. Only the cells between neurons are displayed representing the borders between neurons.

Usually, U-matrix is depicted as a heat map, where lighter colours represent small distances between neurons, i.e. their weight vectors, and darker colours represent neural units that are further apart. An analogy with a landscape is obvious - lighter areas represent valleys whereas darker areas are mountain chains that separate clusters of the neurons apart. Each input data point is mapped to its winning neuron in SOM and all the neurons in the same valley belong to the same cluster. This idea is a foundation of many clustering algorithms that try to automatically determine valleys in the U-matrix and its derivatives and eventually group neurons in a common valley into a cluster. So, let us first review some of them and later give our focus on other approaches towards clustering of SOM.

4.3.1 The U-matrix approach

In 1999, Costa and Netto published a paper that introduces an algorithm for automatic grouping of the SOM neurons using a watershed algorithm [140] from the field of image segmentation to separate regions of the U-matrix [141]. The algorithm was denominated as self-labelling SOM (SL-SOM). Two years later they extended their work with a method that recursively produces a hierarchy of clusters by applying the same SL-SOM algorithm on every cluster of neurons, thus providing the information of cluster structure on different granularity levels [142].

In the same year 2001, Bogdan and Rosenstiel wrote an article about their Clusot algorithm [143]. Again, the basis is the U-matrix, but now enhanced with the so-called hit frequencies of neurons. A hit frequency is simply the number of times a certain neuron is the winning neuron for a data point. So, relative distances between neurons and their hit frequencies are combined together to form the Clusot surface, where clusters are represented by mountains and valleys separate them apart. The authors devised a gradient-based method for automatic cluster discovery. In the following study, Brugger et al. argue this approach has serious limitations, while cluster borders are often unintuitive and the algorithm is very sensitive to one of its parameters [144]. They replace the gradient-based method with recursive flooding algorithm[3], which automatically suggests the number of clusters in the data.

Another attempt to segment U-matrix into clusters with flooding is a semi-automatic procedure by Opolon and Moutarde [145]. Here, the user has to specify the initial point for flooding the U-matrix for each cluster and also the threshold, which controls the flooding limits has to be manually determined. The latter was automatized in the subsequent paper by Moutarde and Ultsch [146], resulting in the U*F algorithm. Moreover, instead of a U-matrix, they built on a U*-matrix, which is basically a rescaled U-matrix in a way that the U-values of neurons with their weights in very dense regions of input data space become even higher and vice-versa [21]. The information about data density around neurons weights is supposed to sharpen the cluster borders and thus improve cluster detection.

In [21], Ultsch introduced the algorithm U*C that is very similar to U*F by its design – they both employ U*-matrix and the flooding algorithm. The novelty is in suggesting that SOM with a huge number of neurons should be used to obtain mean-

[3]Flooding algorithm is one of the watershed-based image segmentation techniques.

ingful information about the cluster structure in the data [147, 148]. Such maps are called emergent SOM (ESOM) with typical number of neurons greater than 4000. Despite superior results being presented, a significantly increased time required for the learning of SOM has to be considered here. Another common issue for all the watershed-based methods is that some neurons of SOM can remain unlabelled – those above[4] the water level when the threshold is met.

In 2006 Samsonova et al. proposed the TreeSOM algorithm that shares the idea of exploiting the U-matrix values for cluster discovery [139]. The algorithm is somewhat similar to U*F, since on each step a neighbouring neuron is added to the current cluster if the distance between that neuron and its neighbour, which is already in the cluster, is smaller than an arbitrary threshold. The final output of the algorithm is a tree that represents the hierarchy of neurons merging. The authors proposed that such a tree is made for different runs of the SOM algorithm with random initialization, and a consensus tree is computed thereafter, which contains only clusters found in the majority of cases. In the final step, the representative SOM that is most similar to the consensus is selected from all the runs. This procedure reminds us of cluster ensemble approach we discuss in details in Chapter 5.

More recently, Newman and Cooper developed a robust and sophisticated algorithm AutoSOME and successfully proved its performance on high-dimensional gene expression datasets [149]. The processing chain in the AutoSOME starts with training SOM from the data and computing the U-matrix, which is cubically rescaled to emphasize the distances between regions of neurons. Then, the SOM neurons are repositioned by the density-equalizing cartogram algorithm that considers the rescaled U-matrix and tries to widen the low-density regions using a diffusion-based approach. Consequently, the neurons that are close together become even closer and gaps between dense regions are wider. The grid of SOM neurons is being treated as a graph and the minimum spanning tree algorithm is applied multiple times to cluster the neurons. Only clusters that occur most frequently, i.e. have their p-value a under user-defined threshold, are taken as the solution. To stabilize output variance, the same approach as in the case of TreeSOM is used - the ensembles with majority voting scheme. The AutoSOME method automatically determines the number of clusters in the data.

In [150], Costa and Yin reviewed common approaches in clustering of SOM with an emphasis on watershed algorithms. They proposed yet another derivation of the U-

[4]In the case of the Clusot algorithm, the neurons under water level are left unassigned.

matrix, called gradient matrix, which is computed as gradient of U-matrix. In [151], Costa applied his SL-SOM method using U-matrix, gradient matrix and newly proposed weighted variants of them both. Each entry in the matrices can be weighted by the corresponding neuron's activity, i.e. its hit frequency. This resembles the Clusot surface in some way. However, experiments were performed on a very limited collection of datasets.

The last representative of clustering methods based on the U-matrix is proposed by Hamel and Brown and we denominated it as SOMStar [138]. Basically, it was meant for visualization purposes only, but we found its interpretation nicely fits into the context of clustering. The story of SOMStar is quite similar to previous ones: first, we obtain the U-matrix from the trained SOM and apply smoothing function on it, e.g. Gaussian kernel smoothing, to homogenize regions of the U-matrix. Then utilize a graph theoretical concept of connected components to find clusters of neurons with simple gradient descent. Neurons act as nodes in an undirected planar graph. Two nodes are connected with an edge, if the gradient between corresponding neurons on the U-matrix is the largest among neighbours. If gradient is zero, there is no edge. The result of this procedure are connected components that look like stars on the U-matrix. It is important to note that Hamel and Brown [138] did not interpret connected components as clusters, but we do not see any obstacle in doing it so.

4.3.2 Other approaches

In the remaining of this section, we survey briefly in chronological order the other proposals of clustering of SOM that are not explicitly based on the U-matrix. The first is the Hierarchical SOM (HSOM), devised in 1992 by Lampinen and Oja [152]. Their two-level scheme is actually a two-layer map of neurons. The first layer SOM is the same as of any other two-level method in this section. The winning neurons are the input for the second-layer SOM that consists of as many neurons as is the desired number of clusters. The authors justified that clusters of arbitrary shapes can be discovered by the two-layered HSOM.

Later on, Murtagh introduced an agglomerative contiguity-constrained clustering method on SOM using the minimal-distance criterion (SOM-CCC-MDC) [153]. It means that on each iteration of the algorithm the closest two clusters are merged together. At the beginning, each SOM neuron forms a cluster. When two clusters are merged, they are replaced by their mean value. The term contiguity-constrained means that we allow merging of two clusters only if there exists a neuron in the first cluster that is a neighbour to any neuron in the second cluster. In this way, the topological order of neurons is considered and exploited. The agglomerative merging process is stopped when only one cluster remains or when a user-defined number of clusters is reached. Murtagh also defined the contiguity-constrained clustering using the minimal-variance criterion (CCC-MVC). Its modification was used in a study of Kiang in 2001, where the minimal-variance criterion outperformed the minimal-distance criterion [136].

Although not the first, but indeed one of the most influential research in the context of clustering of SOM was carried out by Vesanto and Alhoniemi in the year 2000 [117]. For clustering the SOM neurons they employed a well-known *k*-means algorithm [102] and the hierarchical agglomerative clustering methods, i.e. single, complete, and average linkage [50]. Vesanto and Alhoniemi showed that the main advantage of the two-level procedure is a greatly reduced overall running time compared to the clustering of data directly. Another benefit may be the noise removal, as SOM is an approximation of the input space and acts like a filter for outliers. In the mentioned study, the authors investigated the scenario, where the desired number of clusters is known and also the scenario, where the optimal number of clusters is determined using the Davies-Bouldin internal validation index [53]. The accuracy of the obtained partitions is worse, yet comparable to the results of clustering techniques without using SOM. In the following years until present, the research community has been trying to integrate various clustering techniques with SOM to outperform conventional algorithms in terms of stability, accuracy and time complexity. We can also mention a recent study [154], where Chang and Chen compared three two-level clustering methods based on artificial neural networks: SOM, adaptive resonance theory network (ART), and fuzzy ART on the first level. The *k*-means algorithm is applied on the second-level. The authors conclude that two-level scheme benefits from noise removal and low time complexity and that the SOM-based clustering outperforms others.

Work of Wu and Chow [155] is related to Murtagh's and Kiang's as they enhanced the merging process in an agglomerative hierarchical clustering of SOM with the measure of clustering quality CDbw, which stands for *composing density between and with clusters*. Again, only neighbouring clusters are allowed to merge in addition to the local optimization of the CDbw index. Moreover, Wu and Chow suggested that a preprocessing step, which eliminates outliers and noise, is desired before the clustering of SOM.

The success of spectral clustering methods in the field of machine-learning influenced their integration in the framework of SOM as proposed by Taşdemir in 2011 [156]. He demonstrated superior results compared to hierarchical clustering methods as well as the k-means algorithm applied on SOM. In his experiments, he used two implementations of spectral clustering algorithm: the one with manually-adjustable global parameter [157] and its modification that automatically tunes σ locally [103]. The parameter σ is a scale value that determines the pairwise similarities between data points. Furthermore, the spectral methods have high time complexity of $O(N^3)$, where N is the number of input data points. Thus, a vector quantization like SOM is suggested to reduce computational time, yet maintain the level of accuracy [158]. We refer to the spectral clustering of the SOM algorithm as SOMSpec, assuming the variant of spectral algorithm proposed by Zelnik-Manor and Perona [103]. Taşdemir experimented with scenario, where the true number of clusters were known to the clusterers. However, in [103] there is also a description of procedure that is able to automatically determine the number of clusters in the dataset based on the structure of eigenvectors. We use also this version of the spectral algorithm in our experimental comparison in the section 4.5.

The core feature of the spectral methods is a mapping they preform of the input data into the space, where clusters are easier to detect. One criterion for partitioning mapped data is the normalized cut proposed by Shi and Malik [159]. Yang et al. used the normalized cut algorithm on the second level of clustering of SOM [160]. They proposed the following procedure: train SOM and compute the U-matrix[5]; represent the U-matrix as graph with the SOM neurons as vertices and distances between neighbouring neurons as weights of the edges; apply the normalized cut algorithm to partition the graph on clusters of neurons; map each input data point to its winning neuron and label it with the neuron's cluster number. An alternative solution based on the normalized cut is provided by Yu et al. in [161], where the graph weights are computed from the distances between neighbouring neurons and also from distances between neurons and their k-nearest neighbours in the input space, i.e. between neurons weights and data points. The authors report significant gain in computation speed compared to the normalized cut on the original dataset without the SOM layer, as expected, but without decrease in clustering performance.

[5]For consistency, we could place this method also among those based on the U-matrix. But, at the same time, the second-level algorithm relates to the spectral methods that are discussed here.

Recently, Silva and Costa developed two novel methods that implement automatic clusters detection from the trained SOM neurons. They employ a graph cut algorithm [162], and a particle swarm optimization method based on the internal cluster validity index CDbw as a fitness function [163].

4.4 *Gravitational clustering of SOM*

Inspirations for data clustering algorithms design come from various sources – one of them is nature with its laws and mechanisms incorporated. Sir Isaac Newton mathematically formulated the universal law of gravity in 1687, which states that any two mass particles in the universe attract each other with a force proportional to the product of their masses and inversely proportional to the square of the distance between them, so the magnitude of this force between particle **x** and **y** is

$$F = \frac{G \cdot m_x \cdot m_y}{\left(d_E(\mathbf{x}, \mathbf{y})\right)^2}, \tag{4.5}$$

where G is the gravitational constant, m_x and m_y are the masses of particles **x** and **y**, respectively, and $d_E(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between the two particles. Under the influence of the gravitational field around every mass particle, objects move in the space obeying the laws of motion that were also described by Newton. The natural principle of gravity has inspired many researchers in the fields of machine-learning, image processing, optimization, and others, when devising new approaches. In this section we first review some of the most influential works in the cluster analysis that used the universal gravitational law to cluster data. Afterwards, we describe our gravitational algorithm that is strongly coupled with the self-organizing map and works in a two-level scheme, discussed in the previous section.

4.4.1 *Clustering and the law of gravity*

The notion of clustering data with the simulation of a gravity force between data points was firstly proposed by Wright in 1977 [17], although an attempt to discover clusters using forces was published six years before by Coleman [164]. Wright's iterative simulation model computes the total force on every particle from all the other particles and then the particle's new position considering its acceleration and velocity. When two particles are close enough, they merge into one particle with the mass equal to the sum of masses. The maximum distance one particle can travel in each iteration is limited by a user-defined parameter. The algorithm is hierarchical, i.e. it starts with each data point being a mass particle and ends when only one particle remains. Its time complexity is $O(N^3 \cdot D)$, with respect to the number of data points N and their dimensionality D [165]. A user has to manually determine the number of clusters considering the hierarchy of clusters that can be visualized by a tree diagram or a dendrogram.

Oyang et al. extended the Wright's work by exploring orders of the distance-term between particles and showed the promising results when the order is higher than 2 [166]. They modified the simulation model by introducing the air resistance to guarantee the algorithm's convergence. The algorithm is called GRACE and is used in the follow-up study by Chen et al. [167, 168], where they introduced an on-line clustering algorithm GRIN based on GRACE. On-line clustering methods[6] are especially useful for always-growing or huge datasets that cannot be stored in the memory. First, a cluster model is created using only a part of data. When new data sample arrives, it can be put into one of existing clusters or a new cluster is formed. Another on-line algorithm based on the law of gravity is proposed by Gabrijel and Dobnikar in [169] for clustering the state vectors of a recurrent neural-network model.

In 2003, Gomez et al. wrote about the modifications they made to the original Wright's algorithm in order to reduce its time complexity and increase the robustness to noise and outliers [118]. Their randomized gravitational clustering algorithm (RGC) uses simplified equations of movement without velocity vectors. A substantial speed-up is achieved by reducing the number of computational steps when determining the new position of a particle – instead of computing the gravitational force from all the other particles, only the force from a randomly sampled particle is considered. The time complexity thus becomes $O(N \cdot D)$. When the distance between two particles being moved towards each other is less than parameter ε, their fusion is recorded in a separate data structure, but both the particles remain in the system with their masses unchanged, i.e. mass remains unity and is actually not considered by the RGC algorithm. The number of particles remains the same (N) at each iteration, whereas the number of clusters decreases as the particles merge. The authors defined a gravitational constant G as a monotonically decreasing function in time: after every iteration it is reduced by a constant proportion ΔG. So, the particle's movement dynamics decreases each iteration and after a predefined maximum number of iterations the algorithm stops.

[6]On-line methods are called also incremental by some authors.

The clusters that remain in the system are the output of the algorithm and thus, the RGC automatically determines the number of clusters, which is another advantage over the Wright's method. The last step in the RGC algorithm is outliers detection and noise removal – considering the parameter α, clusters with less than α percent of all the particles are removed as they are recognized as outliers or noisy data points. A sensitivity analysis of the algorithms parameters G, ΔG, ε, and α suggests that the performance of the RGC heavily relies on the careful selection of G and ΔG. In [170] an on-line version of the RGC is proposed and discussed.

The simplified GRACE algorithm was used as multi-prototype generator by Long and Jin [171]. Simplification includes elimination of velocity vectors and multi-force attraction calculations – instead, on each step of the algorithm, a pair of particles that will most probably merge, is moved. The mass of particles is considered, but has an inverse effect: heavier objects move slower. The proposed method was used for the recognition of handwritten Chinese characters.

The research community has given a special attention to the novel heuristic optimization procedure called a gravitational search algorithm (GSA) [172]. Rashedi et al. based their work on somewhat modified Newton's equations and it turned out that they actually had not implemented the true gravitational principle. Gauci et al. showed that so-called gravitational force is computed without considering the distance between objects, only mass is taken into account [173]. As the (squared) distance is the essential part of the gravitational principle, the GSA and its derivatives cannot be recognized as gravitational-based, claims Gauci with co-workers. However, the GSA algorithm has become very popular and many extensions have been proposed since. One of them targets also the data clustering and was proposed recently by Dowlatshahi and Nezamabadi-pour [174].

The most recent gravitational clustering algorithm in our survey is the one proposed by Bahrololoum et al. in 2015 [34]. There, an alternative approach is presented, where data points are fixed and only cluster centroids or prototypes are allowed to move according to the forces of gravity applied from the static data points. Before the algorithm starts, a user determines the number of cluster prototypes that equals the number of final clusters. Each data point is assigned to the closest prototype and this prototype's movement is influenced only by the data points assigned to it. The gravitational force is inversely proportional to the p-th order of the distance between a data point and a prototype, where p is a parameter. The gravitational constant decreases linearly with each iteration. The mass of the data points and prototypes is neglected. The algorithm's dynamics is similar to that of k-means due to the moving prototypes that iteratively converge to the cluster's centre of gravity. One advantage over k-means is the mechanism of a local optimum escape that is achieved by adding a random term to the calculation of the movement caused by the gravitational force. At this point we also want to mention another gravitational based algorithm, which resembles to some degree the discussed one, but is not used primarily as a clustering method. In the same year of 2015, Gorman and Valova introduced a self-organizing neural network algorithm GORMANN, similar to SOM, where the neurons are movable agents influenced by the gravitational field around fixed data points [175]. Indeed, there are many differences between this two algorithms, but we find it interesting to mention how similar ideas emerge simultaneously from different research sub-fields.

The gravitational algorithms presented so far are suited only for numerical data in the Euclidean space. Categorical type of data is addressed by Chuang and Chen, who used the correlation between categorical values as attractive or repulsive force, depending on the sign of the correlation [176]. Spherical or circular data, where the distances are measured as the cosine distances instead of Euclidean, was considered by Gomez et al. [177]. Other implementations of the universal law of gravity include gravitational fuzzy clustering algorithm [178], edge detection in images [179], morphological thinning [175], case-based reasoning [180], and SOM visualization enhancement [181] to name only few interesting ones.

4.4.2 The gSOM algorithm

In their study, Gomez et al. summarize that their gravitational algorithm RGC performs adequately well even if up to 80% of data points are removed from the consideration [118]. Moreover, the RGC features the automatic detection of the number of clusters. On the other side, the self-organizing map "needs" a second-level clusterer in order to group the neurons into meaningful clusters without user interaction. For these reasons, an idea to couple the data abstraction method SOM with the gravitational clustering algorithm in a two-level scheme seems to be a perfect match.

Our proposal [15] is based on the algorithm RGC [118] and while keeping many of the RGC features, we redesigned this algorithm in order to integrate it tightly with SOM. In the overview, we represent each winning neuron of SOM as a mass particle that is influenced by the gravity of other particles. Particles are moved around the space and merged together if they are close enough. In this way, the clusters of neurons are eventually formed.

As you probably noticed, we consider only the winning neurons on the second level. The interpolating neurons are eliminated from any further consideration together with the connections to their neighbours, as can be seen on the right-hand side of Fig. 4.1, in order to widen the gaps between the dense map regions that will probably form clusters on the second level of the gSOM algorithm. Vesanto and Alhoniemi [117] proposed this step to make cluster borders more obvious and is indirectly used also by the majority of other methods. Interpolating neurons have in common long distances to the neighbours in average and zero hit frequency. For example, the U-matrix based method Clusot [143, 144] uses the information of distances and hit frequency of the interpolating neurons to separate clusters but does not explicitly remove them.

The identified winning neurons from the first level of the algorithm are now being interpreted as D-dimensional particles in D-dimensional space with a mass equal to unity. The Euclidean distances between these particles are normalized to guarantee that distance between any particle pair is less than or equal to 1. Thus, the dynamics of movements is less dependent on the selection of simulation parameters we define in the following. During iterations of the algorithm, each particle is being moved according to simplified equations describing the Gravitation Law using Newton's Second Law of Motion, as proposed in [118]. The new position of the particles $\mathbf{x}$ and $\mathbf{y}$ influenced by the gravity between them is

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \frac{G(\beta)}{||\mathbf{d}||^2} \cdot \frac{\mathbf{d}}{||\mathbf{d}||}, \tag{4.6}$$

$$\mathbf{y}(t+1) = \mathbf{y}(t) - \frac{G(\beta)}{||\mathbf{d}||^2} \cdot \frac{\mathbf{d}}{||\mathbf{d}||}, \tag{4.7}$$

where $\mathbf{d}$ is a vector pointing towards the particle $\mathbf{y}$, i.e.$\mathbf{d} = \mathbf{y}(t) - \mathbf{x}(t)$, $||\mathbf{d}|| = d_E\left(\mathbf{x}(t), \mathbf{y}(t)\right)$, and $G(\beta)$ is a variable gravitational "constant", which is decreased by the proportion ΔG at each iteration β, following the rule $G(\beta+1) = (1 - \Delta G) \cdot G(\beta)$, in order to avoid the scenario where all the particles will eventually collapse into the same point after sufficiently large number of iterations [118]. When two particles are moved close enough, i.e. $||\mathbf{d}|| < \alpha$, where α is a parameter of the algorithm, they are merged into a single particle with the mass unchanged, which ensures that clusters with a greater density do not affect smaller or less-dense ones. The results of the experiments presented in Section 4.5 prove that such an approach is beneficial. However, our implementation allows a user to enable mass consideration and to experiment with it, but in our tests, we did not observe any improvements, rather the opposite.

The number of particles decreases during iterations, providing the appropriate initial value $G(0)$. This is evident from Fig. 4.2, showing four of 68 iterations of the gravitational algorithm that was applied to the outcome of the first level, presented in the right part of the Fig. 4.1. With red dots we depict the remaining particles and with black lines the connections between them at the beginning of the iterations 2, 10, 30, and the end of iteration 68, when the gravitational algorithm converged.

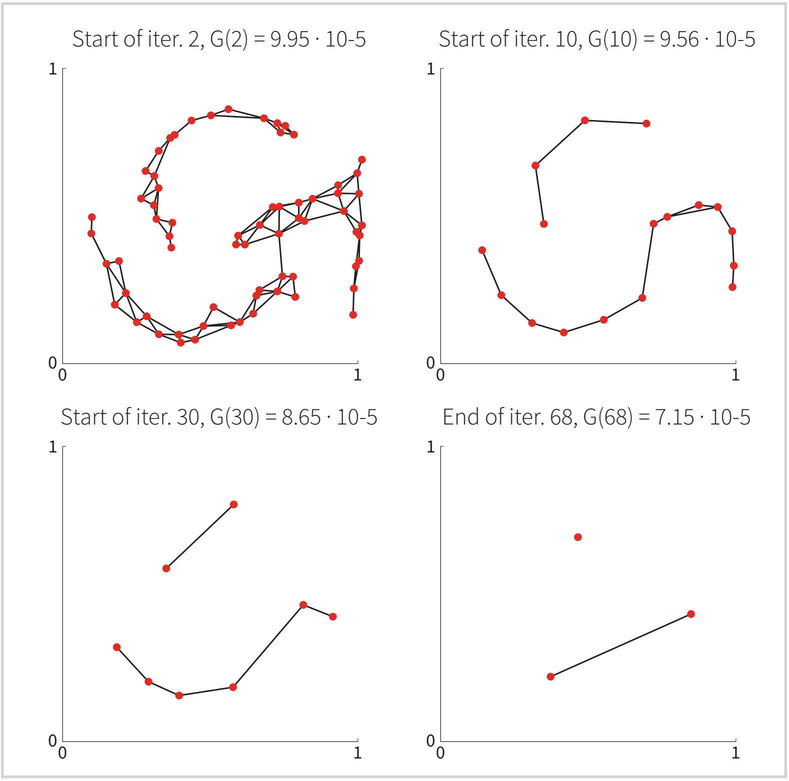


Figure 4.2

Iterations 2, 10, 30, and 68 of the gravitational clustering of SOM, trained on the dataset LCA.

During each iteration of the algorithm, every particle $\mathbf{x}$ from the current set of remaining particles, denoted with $\mathbf{Q}$, is selected once in a random order. Another particle $\mathbf{y}$ is chosen and both $\mathbf{x}$ and $\mathbf{y}$ are moved according to Eq. (4.6) and Eq. (4.7). As the particles are actually neurons from SOM, we can exploit information about the neighbourhood of each neuron to choose particle $\mathbf{y}$. It can be selected in two ways: a) from the set of neighbours of particle $\mathbf{x}$, denoted as $\mathbf{Q}_x$, or b) at random from the whole set $\mathbf{Q}$, including the neighbours of $\mathbf{x}$. With probability p, which is a parameter of the algorithm, we select option a), and with probability $1 - p$ option b). Therefore, the probability that particle $\mathbf{y}$ is chosen from the neighbours of particle $\mathbf{x}$ is

$$P(\mathbf{y} \in \mathbf{Q}_x) = \begin{cases} 0, & \text{if } \mathbf{Q}_x = \emptyset, \\ p + \frac{(1-p)|\mathbf{Q}_x|}{|\mathbf{Q}|-1}, & \text{otherwise.} \end{cases} \tag{4.8}$$

Likewise, the probability that $\mathbf{y}$ is not a neighbour of particle $\mathbf{x}$ is

$$P(\mathbf{y} \notin \mathbf{Q}_x) = \frac{1-p}{|\mathbf{Q}|-1} \left(|\mathbf{Q} \backslash \mathbf{Q}_x| - 1\right) , \tag{4.9}$$

where $\mathbf{Q} \backslash \mathbf{Q}_x$ is a set of the remaining particles without the neighbours of a particle $\mathbf{x}$. When p is large, the movement of a certain particle is influenced more by its direct neighbours, and when p is small, the information about locality is less important. The algorithm stops when G is reduced to a value at which the movements of all the remaining particles are under a particular threshold value ε. In the process of clustering the LCA dataset, shown in Eq. (4.2), iteration 68 was the final one when using $G(0) = 1 \cdot 10^{-4}$, $\Delta G = 0.005$, $\alpha = 0.01$, $p = 0.9$, and $\varepsilon = 10^{-3}$. An alternative stopping criterion is the case when a predefined maximum number of iterations is reached or when only two particles remain in the set $\mathbf{Q}$. The latter implicitly means that we want to split the data into at least two clusters, which is a reasonable assumption. Particles remaining in the set $\mathbf{Q}$ are the final cluster's representatives. Each representative may contain one or more winning neurons and therefore all the data points that belong to these neurons. In this way, for every data point from the input dataset, a final cluster is determined in a process of reconstruction. For example, three clusters were obtained when clustering the LCA dataset, as illustrated in Fig. 4.2.

As the gSOM algorithm is based on the RGC algorithm proposed by Gomez et al. [118], it resembles many of its features. Nevertheless, there are some important differences we want to underline. The first difference is a merging behaviour: on every iteration of the RGC algorithm, there are N particles for they do not merge. The gSOM algorithm merges two particles if close enough, considering the parameter α, and the number of particles decreases over iterations. Secondly, the neighbourhood plays no role in RGC - for each particle a random other particle is selected and then they move towards each other. In contrast, the gSOM is able to utilize the information of the neighbourhood from the SOM to influence the selection of the particles that will move in a particular step; it is done with the parameter p: when $p = 0$, the selection protocol equals that of RGC. The third difference is the stopping criterion: the RGC stops after the predefined number of iterations, whereas the gSOM can stop before reaching this limit if there is no significant movement among particles, which is controlled by ε. Finally, the gSOM algorithm skips the last step of RGC, i.e. the filtering of noise or outliers based on the number of particles in clusters. In other words, data points in the clusters with size less than an arbitrarily selected threshold are eliminated by RGC. We decided to ignore this step due to the fact that SOM already performs noise removal [117, 154]. However, a user can still apply such a filter in a post-processing phase regardless of the chosen method for data clustering. Also note, that in the paper of Gomez et al. there are parameters α and ε too, but with different meaning.

Obviously, the number of discovered clusters depends on a dataset's properties and the parameters' values, particularly six of them: the size of SOM (U), the shape of the SOM grid (rectangular or hexagonal), $G(0)$, ΔG, α and p. Therefore, gSOM determines the number of clusters automatically, without explicitly predefining it, although the user can force the algorithm to stop when a desired number of clusters is reached. Nevertheless, the crucial step, when clustering data with gSOM, is the selection of its parameters, which we address in more detail in Section 4.5. Just a while ago, the authors of the RGC algorithm devised some heuristics to automatically select and adjust its parameters [165]. We found it very interesting and applicable to the gSOM algorithm as well. In the future, we will certainly work towards the update of our algorithm to make it parameter-less if possible.

A clustering algorithm can be classified as partitional or hierarchical, which means it produces a hierarchy of clusters that is visualized as dendrogram. gSOM is a hierarchical algorithm for it is merging neurons that are close enough into bigger clusters. This feature is especially useful for applications in biology, sociology, and behavioural studies as a user gets an insight into clusters' structure at different resolution levels.

Due to a random selection of the particles that will eventually be moved on each iteration, gSOM is a stochastic algorithm. Therefore, it is expected that gSOM, if run several times on the same dataset, produces partitions with a certain degree of diversity between them. This is our main motivation for using gSOM in an ensemble-generation process described in Chapter 5, where the diversity of the ensemble as well as its quality plays an important role in the overall performance of the clustering procedure [20].

4.5 Experimental comparison

In this section we compare some relevant two-level clustering methods with gSOM. Firstly, we compare them by their features and time complexity, followed by a comprehensive empirical evaluation on datasets from various sources. We performed a statistical testing for significant differences of the results, and then we sum up our findings.

4.5.1 Algorithms in comparison

As we have seen in Section 4.3, numerous two-level clustering methods based on SOM were proposed so far. Among them we selected seven representatives to compare with gSOM:

- SOM + k-means (SOMKm) [117],
- SOM + k-means with Davies-Bouldin index to identify the true number of clusters (SOMKm*) [117],
- Clusot* with recursive flooding [144],
- SOM + spectral clustering with local scaling of σ (SOMSpec) [103, 156],
- SOM + spectral clustering with local scaling of σ and eigenvector rotation for automatic number of clusters determination (SOMSpec*) [103],
- SOM + connected components on the U-matrix (SOMStar*) [138],
- SOM + normalized cut (SOMNcut) [160].

The most important difference between listed algorithms is whether they determine the number of clusters K in the dataset automatically or should a user give it as the input parameter. Algorithms that are able to automatically determine K have an asterisk (*) at the end of their names. We have chosen these seven algorithms due to the availability of their source code and to cover a wide range of different approaches towards clustering of SOM. We compare gSOM to other conventional clustering methods not based on SOM in Chapter 5.

In Tab. 4.1 we list all compared algorithms and characterize them using five criteria along with time complexity T. It is an estimation of the second level computational cost without the processing of SOM, which itself is $O(N \cdot U \cdot D)$. N represents the number of input data points, D the number of data dimensions, U the number of the SOM neurons, and K the number of output clusters. Five criteria of comparing the algorithms are:

- A - is the number of clusters automatically detected?
- S - is an algorithm stochastic?
- H - does an algorithm produce a hierarchy of clusters?
- C - does an algorithm consider connections between neurons?
- F - does an algorithm consider neurons' hit frequency?

You may notice that gSOM appears twice in Tab. 4.1. We decided to include two versions in the comparison: one that stops the gravitational clustering when the desired number of clusters K are found (gSOM); and the one that does not need the K specified beforehand (gSOM*).

4.5.2 Datasets

We conducted experiments with the same datasets types as for comparison of internal cluster validity indices in Chapter 3: the synthetic `Complex2D`, the `GENE` and the `REAL` datasets. The details about the `GENE` and the `REAL` datasets are evident from Tab. 3.1 and 3.2, respectively. The important difference is that we used a subset of the whole `Complex2D` family, which is presented in Section 2.3.

As discussed in detail in Chapter 2, we devised a synthetic data generator for two-dimensional datasets with a controllable degree of linearly non-separable clusters. The most important parameters of the generator are: the number of clusters K, the number

Table 4.1

Comparison of the selected two-level clustering methods.

algorithm	A	S	H	C	F	T	ref.
SOMKm	No	Yes	No	No	No	$O(U \cdot D \cdot K)$	[117]
SOMKm*	Yes	Yes	No	No	No	$O(\sqrt{N} \cdot U \cdot D \cdot K)$	[117]
Clusot*	Yes	No	No	Yes	Yes	$O(U \cdot D \cdot N + U^3)$	[144]
SOMSpec	No	Yes	No	No	No	$O(U^2 \log U + U \cdot K^2)$	[156]
SOMSpec*	Yes	No	No	No	No	$O(U^2 \log U + \sqrt{N} \cdot U)$	–
SOMStar*	Yes	No	No	Yes	No	$O(U \cdot D + U \log U + U)$	[138]
SOMNcut	No	No	No	Yes	No	$O(U^2 \cdot D + U^{3/2} \cdot D)$	[160]
gSOM	No	Yes	Yes	Yes	No	$O(U \cdot D)$	[15]
gSOM*	Yes	Yes	Yes	Yes	No	$O(U \cdot D)$	[15]

of data points N, the desired minimal distance between clusters' borders d_{min}, and the desired number of cluster pairs L that are linearly non-separable. All the generated datasets have $N = 500$ data samples that are evenly distributed across clusters and the uniform probability distribution is used to populate each cluster. In total, we created 22 datasets, to which we refer as to `Complex2D` datasets and are a representative part of larger family presented in Section 2.3. In Tab. 4.2 we list all the `Complex2D` datasets together with the parameters of generation process. The values of parameters are chosen to cover some typical cases in cluster analysis:

- small to moderate number of clusters ($K = 2, 3, 6, 8$),
- well separated ($d_{\text{min}} \geq 0.3$) and closely positioned clusters,
- between-cluster interactions at low, medium and high level, with $L = 0, 1$, and 2, respectively.

As you can see, not all the combinations of the parameters are possible or sensible. For instance, if we set $L = 2$, we need at least 3 clusters to fulfil the request for two pairs of linearly non-separable clusters. The same goes with the parameter of desired distance between clusters d_{min}. When we state that a pair or two have to be linearly non-separable, we have to adequately decrease the desired distance between clusters to allow the algorithm to converge. We found suitable values with trial-and-error method.

Table 4.2

Synthetic `Complex2D` datasets used in the experiments, each containing 500 samples in two dimensions.

dataset name	K	d_{min}	L
`Complex2D_K2_D50_L0`	2	0.50	0
`Complex2D_K3_D50_L0`	3	0.50	0
`Complex2D_K6_D50_L0`	6	0.50	0
`Complex2D_K8_D50_L0`	8	0.50	0
`Complex2D_K2_D30_L0`	2	0.30	0
`Complex2D_K3_D30_L0`	3	0.30	0
`Complex2D_K6_D30_L0`	6	0.30	0
`Complex2D_K8_D30_L0`	8	0.30	0
`Complex2D_K2_D15_L1`	2	0.15	1
`Complex2D_K3_D15_L1`	3	0.15	1
`Complex2D_K6_D15_L1`	6	0.15	1
`Complex2D_K8_D15_L1`	8	0.15	1
`Complex2D_K2_D10_L1`	2	0.10	1
`Complex2D_K3_D10_L1`	3	0.10	1
`Complex2D_K6_D10_L1`	6	0.10	1
`Complex2D_K8_D10_L1`	8	0.10	1
`Complex2D_K3_D12_L2`	3	0.12	2
`Complex2D_K6_D12_L2`	6	0.12	2
`Complex2D_K8_D12_L2`	8	0.12	2
`Complex2D_K3_D10_L2`	3	0.10	2
`Complex2D_K6_D10_L2`	6	0.10	2
`Complex2D_K8_D10_L2`	8	0.10	2

4.5.3 *Evaluation protocols*

The nine algorithms we listed in Tab.4.1 are now subjects of experimental comparison using data from various problem domains. We follow the standard procedure for comparison of machine-learning algorithms on multiple datasets using statistical foundations to determine any significant differences between the compared subjects [104–106]. One of the main issues in the evaluation and comparison of the algorithms is the optimization of their parameters. Following the recent comprehensive study on comparison of clustering methods for biomedical data [182], we define three ways to set the values of parameters and we refer to them as *evaluation protocols*. We list and discuss the parameters of the nine algorithms in Section 4.5.4.

The eCVI protocol

The first protocol is called the eCVI protocol and relies on external validity indices. We assume that the perfect clustering solution $\mathbf{C}^T$ for every dataset is given to the evaluator as a ground-truth. Of course, all the clustering algorithms in the comparison are not aware of the ground-truth before they output the solution – through unsupervised learning we want them to reproduce the ground-truth solution. The parameters' configuration that gives the best eCVI score on each dataset is considered as optimal. The eCVI protocol identifies the upper bound of the algorithm's performance – its expressiveness. We measure the degree of matching between clusterer's output and the ground-truth with three external validation indices, namely ARI, AMI, and BCA, we defined in Section 3.6.3. We place more importance on the ARI index due to its repute and recognition by the community. Thus, some detailed results will be given only for ARI, whereas we provide the summaries also for AMI and BCA.

The CV protocol

The second protocol uses the cross-validation technique over datasets to optimize the parameters of an algorithm and we call it the CV protocol. Here we assume that we know the ground-truth for some datasets from a certain domain, i.e. training set of datasets, and we first optimize the parameters on these datasets using the eCVI measure. Parameters' values that yield the best eCVI score on average across all the training datasets are considered optimal. Second, we apply an algorithm on a new dataset with these optimal parameters' values. For illustration, let us consider the `REAL` datasets. There are 15 datasets in the collection. To compute the performance of an algorithm on the `iris` dataset, we consider all the remaining 14 datasets as a training set and search for a parameters configuration that gives the best eCVI score on average. The optimal configuration is then used by an algorithm to produce the partition of the `iris` dataset. Finally, we assess this partition with the eCVI measure to get the performance score. When applying the CV protocol, we assume that datasets from a certain domain are somewhat similar and we can transfer a trained model from one dataset to another.

The CVI protocol

The third protocol employs the internal validity indices for parameters selection, thus it is called the CVI protocol. With this protocol we simulate the situation, where we do not posses external domain knowledge in a form of ground-truth partitions. We run an algorithm for each parameters' configuration and we compute the value of an internal validity index (CVI). The configuration that gives the optimal CVI value is considered as the optimal on a certain dataset. The question is here, which CVI to use for the parameters selection. In Chapter 3 we evaluated 34 indices and identified a set of five CVIs that perform relatively better than others: CON, SEP, SEPmax, I, and DNs. Hence, we employ these indices to estimate optimal parameters setting for each algorithm on every dataset. We compute the average ARI scores over the datasets of different data collections and display them in Tab. 4.3. CVI that gives the best score on average is taken as representative and used for all the following reports, i.e. DNs for the `Complex2D`, and CH for the `GENE` and `REAL` datasets.

Table 4.3

Selection of the internal validity indices for the parameters selection in the CVI protocol. The average ARI scores over the datasets of different data collections are displayed and the best are underlined.

data collection	CH	CON	DNs	I	SEP
`Complex2D`	0.538	0.624	<u>0.695</u>	0.687	0.524
`GENE`	<u>0.139</u>	0.091	0.097	0.089	0.102
`REAL`	<u>0.293</u>	0.191	0.209	0.230	0.204

An experimental procedure overview

The summary of our experimental evaluation is as follows:

1. Fetch a dataset $\mathbf{X}$ from the collection `Complex2D`, `GENE`, or `REAL`. Ground-truth partition $\mathbf{C}^T$ with K clusters is known to evaluator.

2. Optimize the parameters of clustering algorithms using three protocols: eCVI, CV, and CVI. We experimented with algorithms SOMKm, SOMKm*, Clusot*,

SOMSpec, SOMSpec*, SOMStar*, SOMNcut, gSOM, and gSOM*. Methods with * in their names do not receive K as the input parameter. Stochastic methods repeat clustering of the data 30 times for each configuration of their parameters.

3. Apply clustering algorithms to create partitions of the dataset $\mathbf{X}$ using the optimal configuration of their parameters with respect to the defined protocols.

4. Validate partitions using ARI, AMI, and BCA external validation scores with $\mathbf{C}^T$ provided. Repeat steps 1–4 for all datasets.

5. Aggregate multiple runs of the stochastic algorithms using the average of the external validation scores.

6. Compute the algorithms' ranks for every dataset using the average score. Compute the average of ranks over datasets and test a null hypothesis that ranks are equal with the Friedman's non-parametric test [107], which makes no assumptions about the normality of the scores. If Friedman's test rejects the null-hypothesis, make a pair-wise comparison between clustering algorithms using Bergmann-Hommel's post hoc procedure for multiple hypothesis testing [183] with 95% confidence level, i.e. $\alpha = 0.05$.

The reason we used the non-parametric Friedman's test is that "there is no guarantee for normality of classification accuracy distributions across a set of problems" as Demšar pointed out in his paper [104]. To prove our assumptions, we tested the performance scores for normality with Shapiro-Wilk test [184], which is an advisable choice as it is considered relatively powerful against alternatives [185]. For the large majority of cases (>71%) the null hypothesis of the normal distribution of scores across datasets is rejected at significance level of 0.05. We also assessed the scores for the homogeneity of variance assumption with Levene's test [186]. The obtained p-values are below 0.01 for all three datasets collections considering the ARI index and the eCVI and CV protocols, thus we doubt on the validity of the null hypothesis that population variances are equal. Based on the described tests, we conclude that it is not safe to utilize a parametric test, e.g. ANOVA, for the assessment of the experimental results in this study.

4.5.4 Parameters setting

All the algorithms in the comparison are designed as two-level with SOM being the first level processing tool. So, the parameters of each algorithm include the adjustment of the SOM's parameters. We control SOM with two parameters: the map scale factor S and the grid shape g. The scale factor determines the number of the neurons in SOM, which is $U = S \cdot \lceil 5\sqrt{N} \rceil$. We experimented with three values for S: 0.5, 1, and 2. The grid g can be of rectangular (rect) or hexagonal (hex) shape, which means a neuron has four or six neighbours, respectively. The other SOM's parameters are described in Section 4.2 and are not included into a configuration. All nine algorithms have S and g parameters in their configurations, except for the SOMStar*, which was designed only for rectangular grids.

The SOMKm and SOMKm* algorithms are based on the k-means method. We set the number of iterations of the k-means to 100 and we internally replicate the k-means 10 times, with different initialization of the clusters centres. Only the result, which minimizes the sum-of-squares error is kept. SOMKm needs the target number of clusters K to be specified, whereas SOMKm* produces a solution for each number of clusters on the interval $[2, K_{\max}]$. $K_{\max}$ is a function of the number of data points N and the true number of clusters $K_T = |\mathbf{C}^T|$, bounded by 25 or by $K_T + 5$ if K_T is larger than 25. To be explicit

$$K_{\max} = \max\left\{\min\left\{\left\lceil \sqrt{N} \right\rceil, 25\right\}, K_T + 5\right\} . \tag{4.10}$$

Using the Davies-Bouldin internal cluster validation index, the optimal solution predicted by this index is returned as the output.

The Clusot* algorithm is based on recursive flooding and needs three parameters to adjust, namely the initial water level θ_0, the fraction of the Clusot surface to be flooded θ, and the resolution `res` of the grid on which the Clusot surface is computed. In the agreement with the experimental setting in the [144], we used the following values of each parameter: $\theta_0 = [0, 0.2, 0.4, 0.6]$, $\theta = [0.1, 0.3, 0.5, 0.7, 0.9]$, and `res` $=$ $[0.01, 0.05, 0.1, 0.25, 0.5]$. Only the configurations, where $\theta_0 < \theta$ are meaningful and considered.

The parameter `kNN` of the SOMSpec and SOMSpec* algorithms, which controls the local scaling of the σ value was set on a range of values $[1, 2, 3, 5, 7, 9, 11, 15]$. Moreover, the SOMSpec* algorithms automatically determines the number of clusters by rotating the eigenvectors of the data and needs a range of numbers to be specified for testing. As with SOMKm*, we define this interval to be $[2, K_{\max}]$, where $K_{\max}$ is the same as in Eq. (4.10).

As it is evident from the description of the SOMStar* algorithm in Section 4.3, the U-matrix of SOM is smoothed with a Gaussian kernel before further processing. The parameter σ controls the bandwidth of the kernel and the following values are used: $[0, 0.5, 1, 1.5, 2, 2.5, 3]$. The SOMNcut algorithm does not need additional parameters, except for the number of clusters.

In Section 4.4 we presented the gSOM algorithm with six parameters that need to be set, i.e. the initial gravitational constant G, the decay ΔG of the G during iterations, the merging distance α, the probability p of selecting a neighbouring particle, the threshold of moving ε, and the maximum number of iterations. Our experiments with gSOM, partly published in [15, 18], reveal that the majority of the parameters can have default values. In the experiments in this thesis, we declare the following defaults: $\alpha = 0.01$, $p = 0.9$, $\varepsilon = 10^{-3}$, and maximum of 500 iterations. However, G and ΔG have the biggest impact on the performance and thus are the subjects of a searching procedure, where the following sets of values are being tested for gSOM: $G = [10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 3]$ and $\Delta G = [0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5]$. Remember, gSOM stops when the number of clusters reaches the predefined number K and gSOM* does not. So, when experimenting with gSOM, we allow ΔG to be zero, which means the gravitational force does not decrease in time and if we did not stop the simulation, all the particles would eventually collapse into one. Therefore, we modify the vector of ΔG for gSOM*: $\Delta G = [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]$. As we take zero away, we add one to widen the search space. Values for G remain the same. We reach a total of 420 different configurations of parameters' values for gSOM, gSOM* and Clusot*, which is our upper limit for experiments to remain feasible.

While developing the gSOM algorithm, we noticed its behaviour is sometimes unstable due to the high degree of built-in randomness. It means that in a series of runs we can experience some poor results. To stabilize gSOM's behaviour we employ similar technique as with KM: each run of the gSOM algorithm makes 5 internal replicates and only the best solution is considered as the output of the algorithm. The best solution is defined as the one that optimizes the internal validation index. Again, we are facing a dilemma, which CVI to use. We performed experiments with the CH, CON, DNs, I, and SEP indices and select the one that is the best on average over all datasets regarding each dataset collection. The averages are shown in Tab. 4.4. The following CVIs are thus used for all reports: CH for the `Complex2D` and `REAL`, and CON for the `GENE` datasets. Note, that differences between CVIs are not so big and we believe that our decision has minor impact on the presented results.

Table 4.4

Selection of the CVI for the stabilization of the gSOM results. The average ARI scores over the datasets of different data collections are displayed and the best are underlined.

data collection	CH	CON	DNs	I	SEP
`Complex2D`	<u>0.931</u>	0.922	0.921	0.925	0.901
`GENE`	0.327	<u>0.328</u>	0.317	0.327	0.307
`REAL`	<u>0.382</u>	0.368	0.345	0.370	0.351

We list the optimal parameters configurations considering the eCVI and CVI protocols for all datasets and algorithms in Appendix A. For instance, in Tab. A.4 we display the optimal parameters that are selected by the eCVI protocol for the `Complex2D` datasets using the ARI score. In the last row there are configurations that are best on average over all datasets and can be thus considered as the suggested defaults if nothing is known about a dataset except of its domain. Similarly, Tab. A.5 and Tab. A.6 show the optimal configurations for the `GENE` and `REAL` datasets. The optimal parameters regarding the CVI protocol are listed in Tables A.7, A.8, and A.9. Let us summarize some of our findings considering the eCVI protocol and far from being exhaustive:

- The optimal size of SOM (scale factor S is directly proportional to the number of neurons U) varies greatly among algorithms and datasets collections.
- Fewer variations are noticed considering the SOM grid shape g, although some are still very distinctive. The gSOM*, gSOM, SOMSpec, and SOMSpec* methods, for instance, strongly prefers rectangular grid when clustering the `Complex2D` datasets. This is not the case when dealing with the `GENE` and `REAL` datasets.
- The Clusot* parameters θ_0 and θ have a tendency to be of rather small values regardless of the dataset's type. The optimal value for the θ_0 is often zero, which

means that no flooding in the start of the recursive process is favourable.

- The SOMSpec algorithm, and to some degree also SOMSpec*, prefers smaller values of the `kNN` parameter. Surprisingly, for many cases, the value of 1 is the best. This means that in computing the similarities from the distances between SOM neurons, only one nearest neighbour of a neuron is considered.

- We observe that the smoothing strength σ of the SOMStar* algorithm is set to zero in many of the best configurations in the context of the `GENE` datasets. This means no smoothing of the U-matrix.

- The behaviour of the gSOM's parameter G is again very interesting. When clustering low-dimensional `Complex2D` datasets, low values are preferable. On the other hand, for high-dimensional `GENE` datasets, high values of G are more positive. This is somewhat expected, as the data gets sparser in higher-dimensional spaces. Thus, stronger gravitational force is needed to pull the data samples together and to cluster them.

We had to limit ourselves in searching for the best parameter's configuration, and as the majority of the best performing parameters' values are below their extreme levels, we believe our sampling of the search space is decent.

4.5.5 *Results*

We discuss obtained experimental results for each type of datasets separately and then try to generalize our findings on the entire study. Our main tool for the results' visualization is a box plot that displays the summary statistics of the algorithms' performance along with the average ranks, where the significant pair-wise differences between algorithms are indicated. Omnibus Friedman's test rejects the null hypothesis of equal ranks with p-value below 0.008 in the worst case for all the datasets considering the ARI score. The only exception, when Friedman's test does not reject the null hypothesis is with the `REAL` datasets that are validated using the AMI score and the CV protocol. In this chapter, we included only the results based on the ARI score – see Appendix A for additional plots using the AMI and the BCA external validation indices.

`Complex2D` *datasets*

The results on the `Complex2D` datasets are visualized in Fig. 4.3 with three panels corresponding to three protocols for parameters optimization, namely eCVI, CV, and CVI. Each panel is composed of box plot on the right and the plot showing the significant differences between algorithms on the left side. The comparing algorithms are sorted by the average rank over datasets, which is shown on the very left – the lower the rank, the better the algorithm is supposed to be. Those algorithms that are not recognized as significantly different by their performance are connected with red bar. The box plot on the right shows the distribution of scores for each algorithm. The median score is marked with a vertical black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black points. So, we interpret the top-positioned panel as follows. We considering the best configuration of the parameters for each dataset using the average ARI score across multiple runs of the algorithms. Here we see two large groups of algorithms that perform significantly different: the first one contains SOMSpec, gSOM, SOMSpec*, and gSOM*; the second consists of SOMStar*, SOMKm*, and Clusot*. For this two groups it is true that all the methods from the first group outperformed all the methods in the second one. The SOMKm and SOMNcut algorithms are in between, meaning that SOMKm outperforms all the algorithms from the second group and SOMNcut is worse than all the algorithms in the first group, but we cannot tell whether SOMKm is significantly better than SOMNcut or not, so we do not include the two in the neither group. When we interpret the results of the eCVI protocol, we have to be very careful and bear in mind that we measured the upper limit of the algorithms' capability, their expressiveness. This is not their effective performance in a real-world situation, unless we are lucky enough and guess their optimal parameters' values. So, we should better not put too much importance on the top-performers, rather we can identify the methods that have low potential on certain kind of data.

In addition to the eCVI protocol, the CV and CVI protocols simulate the situation, where we optimize the algorithms' parameters without the access to the ground-truth of data. Here, SOMSpec, SOMKm, and gSOM prove to be significantly better than SOMKm*, SOMStar*, and Clusot*. The scores are lower with the CV protocol than with eCVI or CVI in general – we assume this is because the cross-validation over datasets requires very strong correlation between datasets, otherwise the parameters' estimation on a subset of datasets fails when they are transferred to the new data. The SOMSpec and gSOM algorithms are close competitors and SOMSpec often achieves higher average score and rank than gSOM. However, the time complexity of the SOMSpec algorithm is considerably higher than that of gSOM as can be seen in Tab. 4.1, i.e. $O(U^2 \log U + U \cdot K^2)$ for SOMSpec versus $O(U \cdot D)$ for gSOM, where we do not include a time needed for the training of SOM.

Quite similar results are obtained using the AMI and the BCA validity measures, with some minor changes in algorithms' ordering – the corresponding charts are presented in Figs. A.1 and A.4 in the Appendix A. Moreover, in Tab. A.1 we display how many clusters the algorithms discovered in their best run, when the true number K_T is not given as the input parameter. One may think that counting the occasions when the discovered number of clusters matches with K_T for a given dataset, should characterize an algorithm's performance. We argue this is not a sufficient measure of success, since an algorithm could find exactly K_T clusters, but these clusters might be in total disagreement with the target ground truth partition. Therefore, we rely rather on eCVIs, i.e. ARI, AMI, and BCA, which measure the agreement between ground-truth and the output of a clustering algorithm. If the eCVI indicates good performance of an algorithm and the estimated number of clusters also matches with the expected K_T, we may conclude an algorithm resembles well the gold standard or ground-truth. Nevertheless, we observe that SOMSpec* is the most consistent with K_T considering the eCVI protocol, followed by gSOM*, SOMKm*, SOMStar*, and Clusot*. When the algorithms' parameters are optimized using cross-validation (CV protocol), gSOM* achieves the best matching rate with K_T. For the CVI protocol, SOMSpec* and gSOM* both find K_T number of clusters on 12 datasets.

GENE *datasets*

The datasets with gene-expression profiles are known for their uncommon shape: only a few data samples in ultra high-dimensional space. Each gene acts as one dimension and there are typically hundreds of genes measured in an experiment. So, due to the sparsity of the data, to which we often refer as the curse-of-dimensionality, it is very challenging to cluster genetic data with high accuracy. As expected, the ARI scores for the GENE datasets are substantially lower than for two-dimensional synthetic

Complex2D datasets. The statistical pair-wise comparison among algorithms reveals fewer significant differences. For example, if we consider the eCVI protocol, the top-positioned gSOM method performs better only than SOMNcut and and SOMKm* as we show in Fig. 4.4. We find it interesting that the ordering of the algorithms is so different when switching from eCVI to the CV or CVI protocol. Measuring the expressiveness with eCVI, we discover that SOMKm, SOMNcut, and SOMSpec have their mean score values rather low. But, considering the CV and CVI protocols, they are placed on the top of the list. We see one possible explanation in fact that GENE datasets are gathered from various sources and it is hard to successfully guess good parameters using CV. The same is true with the CVI protocol as we see in Chapter 3 that the correlation between CVIs and eCVIs is low on this domain.

A look at the results based on AMI and BCA in Figs. A.2 and A.5 brings us to a similar conclusions. The gSOM* algorithm has the second highest number of matches between the estimated numbers of clusters and the true values for eCVI and CVI protocols, as can be seen in Tab. A.2. The matching rate of gSOM* is very low with the CV protocol, whereas Clusot* correctly guesses the number of clusters in more than a half of cases.

REAL *datasets*

Fifteen real-world, non-genetic, datasets from various sources are now a playground for methods' comparison. In Fig. 4.5 the average ranks and the ARI score distribution over REAL datasets are presented. Curiously, gSOM* gets better rank than gSOM when measuring the expressiveness, which we attribute to the stochastic nature of both algorithms – however, the mean score of gSOM is slightly higher than that of gSOM*. The differences in ranks against complementary two-level methods show that gSOM and gSOM* expressiveness is significantly higher than SOMKm*'s. Considering the CV and CVI protocols, gSOM's and gSOM*'s performance is not significantly different of others.

Additional results are placed in Appendix A, with AMI and BCA based scores in Figs. A.3 and A.6. We also enclose the information about the number of estimated clusters in Tab. A.3. We conclude that the SOMSpec* and the gSOM* methods are the most successful in guessing the number of clusters in data.

4.6 Conclusion

The main subject of this chapter was the gravitational clustering algorithm based on the self-organizing map. It is a two-level clustering approach we proposed to tackle the challenges in different aspects of data complexity. We reviewed and pointed out the most influential works in the context of two-level cluster analysis based on SOM and motivated the usage of yet another principle to group the neurons of SOM in the meaningful clusters – we tightly integrated the gravitational clustering algorithm proposed by Gomez et al. [118] with SOM and thus developed a unique algorithm that is able to automatically estimate the number of clusters[7] in the data and has linear time complexity with respect to the number of neurons in SOM. Moreover, the gSOM produces a hierarchy of clusters, which could be a helpful tool for a data-miner to better grasp the structure of the data.

Our contribution in the field is the gSOM algorithm and its comparison to the seven complementary two-level algorithms in a qualitative and quantitative way. The latter is done via extensive experiment that included various data types, from synthetic datasets that model complex clusters boundaries and their linearly non-separable interactions, to the real-world data with gene expression measurements and other benchmark datasets well-known to the machine-learning community. We measured the algorithms' performance under three different protocols for the optimization of the parameters.

Based on the empirical evidences from our experiment, we conclude that there is no significant difference between the performance of the two variants of the gSOM algorithm, i.e. the one that knows the number of clusters and the other that has no clue about it and tries to estimate how many clusters are in the dataset. However, knowing the true number of clusters naturally results in higher average score. Our main finding is that the gSOM algorithm performs well, especially on genetic data, where it shows high expressiveness. The main advantage over its closest competitor, the SOMSpec method, is a substantially lower time complexity and thus better scalability.

One of disadvantages of the gSOM algorithm is sensitivity to its parameters, especially the initial gravitational constant G and its decay on every iteration ΔG. Fortunately, the authors of the gravitational clustering algorithm [118], which inspired our

[7]We also included in our experiments a variant of gSOM that exits the simulation of the gravity when the desired number of clusters is reached.

work, recently presented a heuristic approach towards the parameters value estimation without the intervention of the user [165]. Our plan in the future is to adapt and integrate their findings into the gSOM method to fully automatize it.

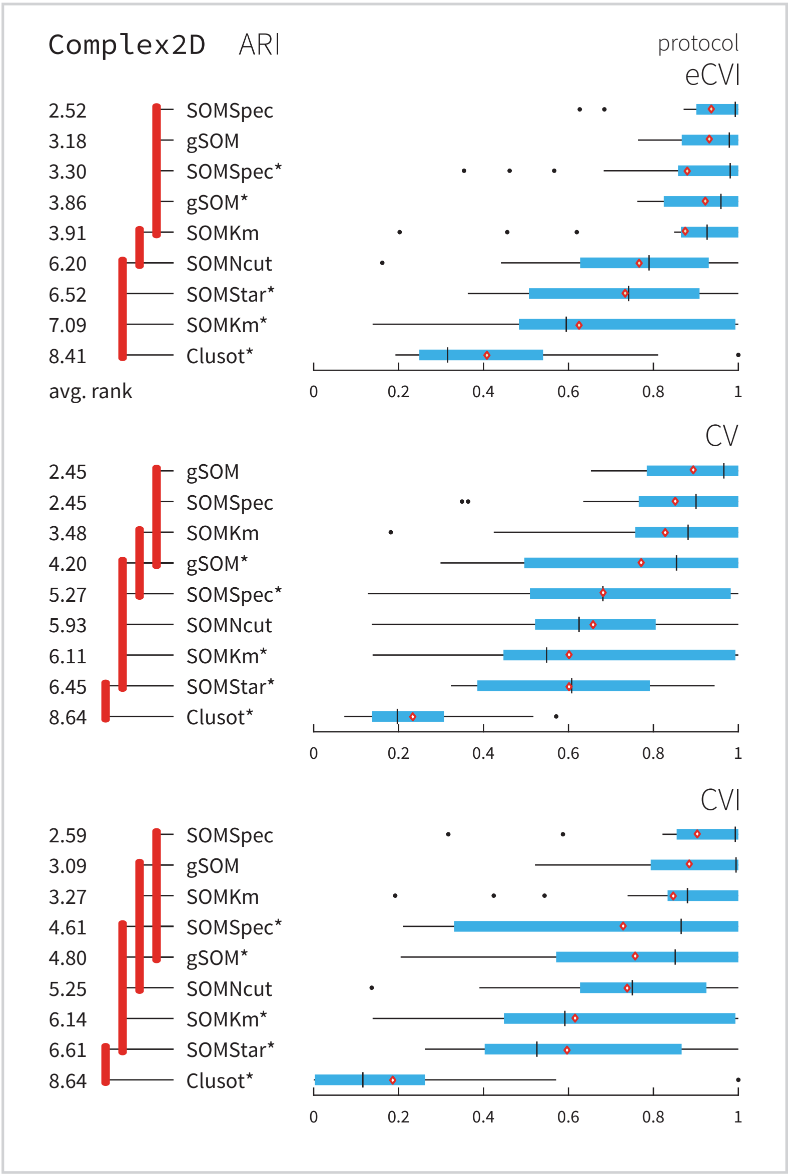


Figure 4.3

Average ranks of algorithms based on the ARI score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

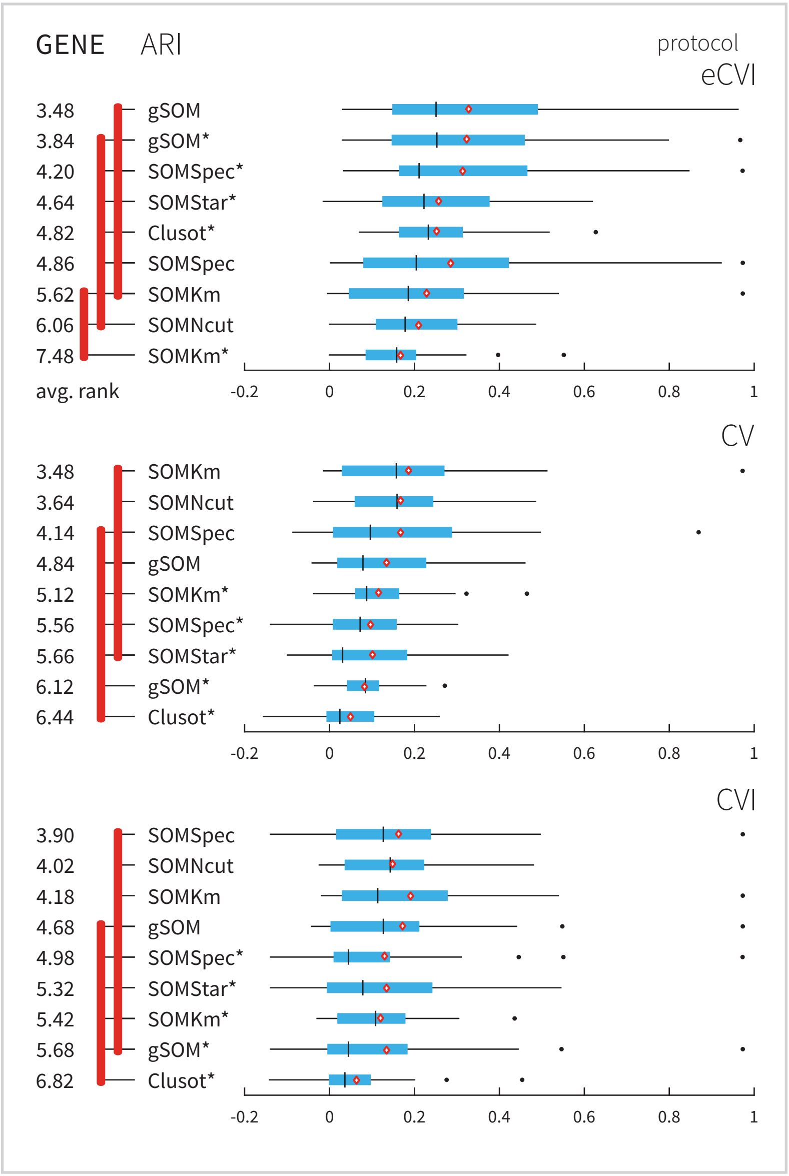


Figure 4.4

Average ranks of algorithms based on the ARI score across all the GENE datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

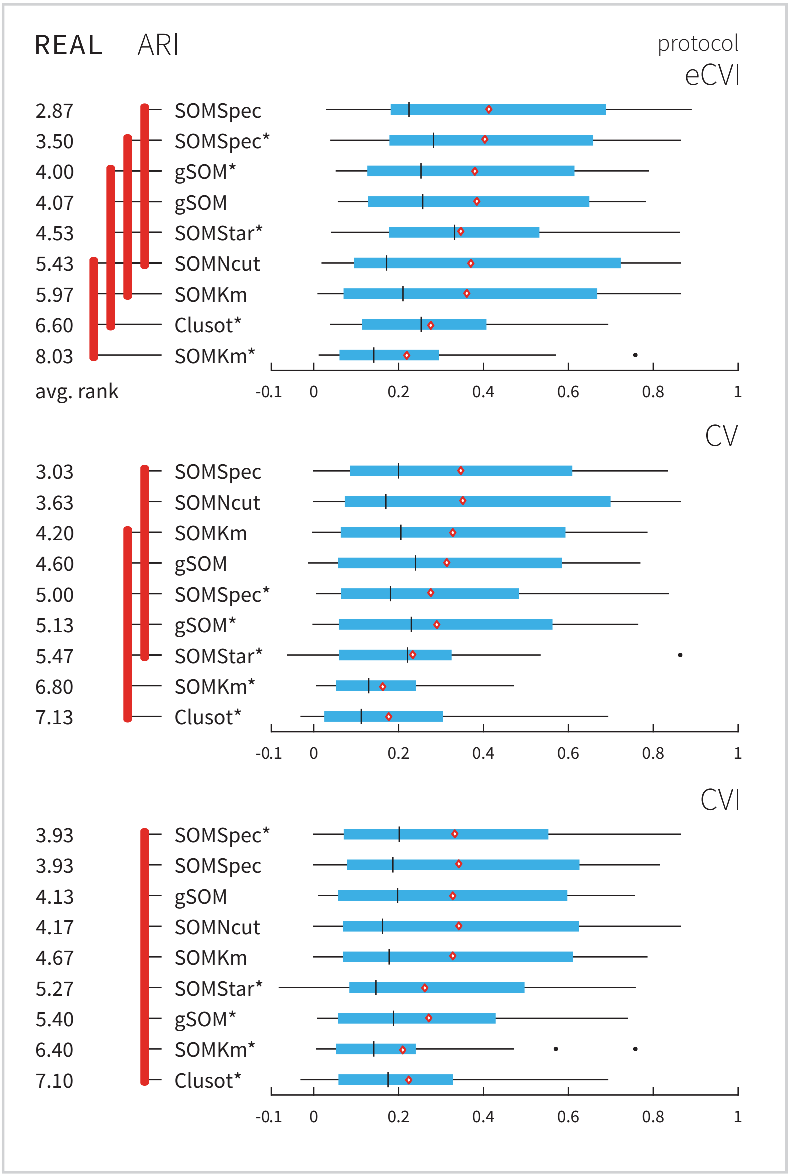


Figure 4.5

Average ranks of algorithms based on the ARI score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

5

Weighted Cluster Ensemble

5.1 Introduction

It is very hard to provide a unified framework for clustering methods development, although some attempts have been made recently towards it [187]. The stated issue is known as an impossibility theorem [8], proving the non-existence of a clustering algorithm that would satisfy all the common assumptions of a good clustering. This is the main reason why a plenty of clustering algorithms has been proposed so far, using various theories and approaches [4]; from probabilistic methods [1], spectral analysis [103, 188], artificial neural networks [116], information theory [189], and kernel functions [190] to combinatorial search techniques and fuzzy logic [5], to name only a few.

Solving the problem of supervised classification using a combination of multiple classifiers is well studied and well known approach [9] that has inspired the research of incorporating this idea in the unsupervised manner. An increasing interest in cluster ensemble[1] methods has been witnessed during the last 15 years [191] and proposed techniques show noticeable advantages over single-clustering algorithms in the sense of stability and robustness [88], as well as improved quality of the results [192, 193]. For more comprehensive overview and survey of cluster ensemble techniques see [194–197].

Clustering ensembles became an interesting topic in the very beginning of the third millennia. In the 2001, Fred published her first results of experimenting with combining multiple partitions into consensus solution [198]. She proposed a voting scheme called *evidence accumulation clustering* (EAC) [199], where each pair of data points gets as many votes as many times it appears in the same cluster among multiple partitions. This idea has been generalized into different voting mechanisms based on *co-occurrence of data object* since then and represents one of two main streams of the cluster ensemble field [197]. The second one is based on searching for the *median partition*. Various optimization methods are utilized to find the partition, which is supposed to be the best representative of the ensemble of partitions given a similarity measure between partitions. For this purpose the external cluster validity indices are used. Our work extends the evidence accumulation principle and therefore we will not address the median partition methods in further detail for now. Nevertheless, we acknowledge that

[1]Here, the word ensemble has two meanings; we can refer to it as a set of partitions or as a set of methods that generate these partitions.

the future research directions include the consideration of the latter.

A year after Fred's paper, Ghosh wrote about advantages of multiclassifier systems and correctly predicted the penetration of this paradigm in the field of unsupervised learning [191]. Together with Strehl they devised three consensus functions, namely the cluster-based similarity partitioning algorithm (CSPA), the hypergraph partitioning algorithm (HGPA), and the meta-clustering algorithm (MCLA), that have been widely accepted by the community and represent the foundation of many subsequent studies of cluster ensembles [192]. The other works that strongly influence the development of the field in the early years are those of Fischer and Buhmann [200], Monti et al. [201], Fern and Brodley [202, 203], Dudoit and Fridlyand [204], and Kuncheva and Hadjitodorov [205]. All these studies describe the process of data clustering with ensembles as a two-step procedure:

1. generation of cluster ensemble,
2. consensus function.

The cluster ensemble $\mathbf{P}$ consists of M partitions $\mathbf{C}_m$ made on a dataset $\mathbf{X}$ using single-clustering algorithms[2], so $\mathbf{P} = \{\mathbf{C}_1, \mathbf{C}_2, \dots \mathbf{C}_M\}$. It is shown that diversity and accuracy of partitions within the ensemble greatly influence the consensus solution quality [202, 205–207]. Some researchers advocate generating as diverse ensembles as possible, yet others advise more moderate level [208]. However, they agree the ensemble should contain variation between solutions if we want to gain some novelty from it. There are many ways to ensure the diversity of the ensemble:

- with manipulation of the input dataset $\mathbf{X}$ using
  - subsampling of data objects or data features/dimensions,
  - projections of data to subspaces, or
  - different representations of the problem domain;
- with manipulation at the level of clustering methods. This includes
  - a set of various clustering methods, i.e. heterogeneous ensemble, or

[2]We addressed the single-clustering algorithms in Chapter 4, where we discussed the gSOM algorithm.

- one method with different settings of its parameter values, i.e. homogeneous ensemble.

Naturally, both techniques can be combined. Here we study the random subsampling of data features in combination with homogeneous type of ensemble. We will discuss it in more details later in Section 5.4. A large majority of the studies we examined, experiments with the homogeneous ensembles, e.g. [10, 88, 199, 203, 205, 209–212] to name only few – nevertheless, we acknowledge that an ensemble generation based on different clustering algorithms may be interesting to exploit as in [13, 192, 213–215].

After the ensemble of partitions $\mathbf{P}$ is generated, we have to compute the final solution $\mathbf{C}^{\mathbf{P}}$ using consensus function. In general, we cannot expect that consensus partition will outperform each individual member of the ensemble. But on the other hand, we suppose the final solution will be robust with respect to ensemble members of bad quality and will be better than ensemble members on average. To help the consensus function to achieve its goal an additional step was proposed just after ensemble generation: weighting of partitions. Different authors give different names for this step; we adopt nomenclature from Vega-Pons et al. [10], where they called this step the Partition Relevance Analysis step (PRA). The basic idea is to validate each ensemble member $\mathbf{C}_i$ and give it a weight w_i that resembles its quality or relevance. Thus, partitions of bad quality, i.e. noisy ones, will eventually get small weights, which means they will contribute less to the consensus solution. Therefore, we hope the consensus function would find more easily a good solution. Of course, one has to adapt a consensus function to consider assigned weights. As far as we know, Duarte et al. were the first that proposed the weighted cluster ensemble approach in 2005 [13]. They built on the evidence accumulation framework and introduced a weighting of partitions in the ensemble using the internal cluster validity indices. This is also a starting point of our research work that we explain in more details in the next two sections, where we introduce an enhancement to the PRA step. We extensively evaluate our proposal in Section 5.4.

Before we move on, let us acknowledge that a field closely related to the weighted ensembles is the *cluster ensemble selection* [212, 216–219]. With selection we mean that partitions are selected from the ensemble and only the selected ones are considered by consensus function. So, weighting is somehow a generalization of selection, as a weight $w_i = 0$ usually means that the partition $\mathbf{C}_i$ is not considered by consensus function at all and this equals the decision not to select this partition. Weighting and selection of partitions from the ensemble utilize similar techniques, however we will not address the selection schemes here.

5.2 Weighted evidence accumulation

In this section we first explain the principle of evidence accumulation approach in cluster ensembles and show its weighted derivations. Then, we focus on weighting of ensemble using internal cluster validity indices and motivate our contribution to enhance the partition relevance analysis.

The core of the evidence accumulation approach is matrix $\mathbf{S}$ that consists of $N \times N$ entries, each representing the similarity between a pair of data points $\mathbf{x}_i$ and $\mathbf{x}_j$ based on the cluster memberships in the ensemble of M partitions $\mathbf{C}_m \in \mathbf{P}$:

$$\mathbf{S}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \lambda_{ij}^{\mathrm{EA}}(\mathbf{C_m}) \,, \tag{5.1}$$

where

$$\lambda_{ij}^{\mathrm{EA}}(\mathbf{C_m}) = \begin{cases} 1, & \text{if } \exists \mathbf{c} \in \mathbf{C}_m \colon \mathbf{x}_i \in \mathbf{c} \wedge \mathbf{x}_j \in \mathbf{c} \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

So, the matrix $\mathbf{S}$ measures how many times a certain pair of data points co-occurs in the same cluster. This matrix is then regarded as a similarity matrix of data points and is clustered into desired number of clusters using conventional single-clustering methods like the agglomerative single-linkage algorithm (SL).

There are many variations of consensus functions based on the evidence accumulation principle with respect to clustering method that is used to process the $\mathbf{S}$ matrix. For instance, the popular CSPA function [192] considers $\mathbf{S}$ as an adjacency matrix of a graph, where data points $\mathbf{x}$ are the vertices and $\mathbf{S}_{ij}$ values the weights on the edges between the vertices. The graph is then partitioned using the hyper-graph partitioning algorithm METIS.

Moreover, recently devised *divisive clustering ensemble with automatic cluster number* (DICLENS) by Mimaroglu and Aksehirli [220] also relies on the counting of the data points co-occurrence in the clusters. They measure a similarity between clusters as the average similarity between data points in those clusters, which is again computed as evidence accumulation. Then they build a graph, where clusters correspond to vertices and similarities between clusters correspond to weights on the edges. A similarity-based minimum-cost spanning tree (SMST) is constructed on the graph to provide a starting point for automatic discovery of final clusters along with their number. SMST's edges are being removed iteratively starting with the edge with the smallest weight. The remaining connected components on each iteration are meta-clusters, i.e. clusters of clusters. Each meta-cluster becomes a cluster in the final output by the majority voting procedure. On each iteration the quality of the output clusters is computed by the means of cluster compactness and separation between clusters, which is founded on the evidence accumulation. When all the edges of the SMST are removed, we select the result from the iteration that maximizes the quality function. In this way, the DICLENS algorithm automatically determines the number of clusters. We spent some time describing the DICLENS method because we propose its minor extension. In the case, we do not have any idea of how many clusters we expect in the consensus partition, the described automatic procedure is at most welcome. However, if we want the method to find exactly K clusters, we could do it by considering only those edges removal that produce exactly K connected components, i.e. meta-clusters. Among all the possibilities we select the one with the highest quality score.

Now, let us proceed with the presentation of the modifications of the evidence accumulation approach using weights. We consider two types of weighting the similarity matrix $\mathbf{S}$: those based on the cluster properties, and those that use the cluster validity indices to measure partitions' relevance. In 2009, Wang et al. proposed an enhancement to evidence accumulation by taking into account also the clusters' sizes of the ensemble partitions [221]. Their formulation of the similarity matrix $\mathbf{S}$ is called the *probability accumulation* (PA) and is defined as

$$\mathbf{S}_{ij} = \frac{1}{M}\sum_{m=1}^{M} \lambda_{ij}^{\mathrm{PA}}(\mathbf{C_m}), \tag{5.3}$$

where

$$\lambda_{ij}^{\mathrm{PA}}(\mathbf{C_m}) = \begin{cases} 1, & \text{if } i = j \\ \frac{1}{1+\sqrt[D]{n_k}}, & \text{if } i \neq j \text{ and } \exists \mathbf{c}_k \in \mathbf{C}_m\colon \mathbf{x}_i \in \mathbf{c}_k \wedge \mathbf{x}_j \in \mathbf{c}_k \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

Here, D is the dimensionality of the data points $\mathbf{x} \in \mathbf{X}$ and n_k is the number of data points in the cluster $\mathbf{c}_k$. The study using this formulation reports better results than using the $\lambda_{ij}^{\mathrm{EA}}$. The improvement is supposed to be due to the finer resolution of the matrix $\mathbf{S}$ that incorporates more information about the partitions. Similar idea lies behind the work of Lourenço et al., where they proposed mathematically well-founded probabilistic approach to the EAC paradigm using the optimization of a Bregman divergence between the measured co-occurrence matrix $\mathbf{S}$ and the co-occurrence probabilities that two specific data points are expected to be clustered together parametrizing the Binomial random variables [215, 222]. This leads to a model, where consensus partition is not crisp but in which data points are assigned to a cluster with a certain probability. Moreover, they proposed also a weighting mechanism that tries to optimize the contribution of each partition in the ensemble to the consensus partition $\mathbf{C}^{\mathbf{P}}$ [223]. The weights are regarded as probabilities and to avoid a trivial solution, where one partition in ensemble has weight 1 and others are zero, two regularizations of probability distribution are proposed using a restricted simplex or $l2$-norm.

Furthermore, Vega-Pons and Ruiz-Shulcloper proposed an alternative solution taking into account also the number of clusters in the partitions and the similarity measure between data points [224]. There they defined the *co-association significance* that we denote here as WA:

$$\mathbf{S}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \lambda_{ij}^{\mathrm{WA}}(\mathbf{C_m}) \,, \tag{5.5}$$

where

$$\lambda_{ij}^{\mathrm{WA}}(\mathbf{C_m}) = \begin{cases} \dfrac{K_m / \max_{\mathbf{C}_u \in \mathbf{P}} K_u}{n_k / \max_{\mathbf{C}_u \in \mathbf{P}, \mathbf{c}_v \in \mathbf{C}_u}} \cdot \dfrac{\mathrm{sim}_m(\mathbf{x}_i, \mathbf{x}_j)}{\max_{\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}} \mathrm{sim}_m(\mathbf{x}_i, \mathbf{x}_j)}, & \text{if } \exists \mathbf{c}_k \in \mathbf{C}_m \colon \mathbf{x}_i \in \mathbf{c}_k \wedge \mathbf{x}_j \in \mathbf{c}_k \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

Here, K_m is the number of clusters in the partition $\mathbf{C}_m$ that is normalized by the maximum number of clusters over all the ensemble partitions. Next, n_k is the number of data points in the cluster $\mathbf{c}_k$, again normalized by the maximum number of data points in any cluster among ensemble partitions. The similarity measure used for generation of the m-th partition is denoted by sim_m and quantify the similarity between two data points. More common is the usage of dissimilarity measures when clustering data, like Euclidean distance d_E. If so, the similarity is computed from the distance as

$$\mathrm{sim}_m(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + d_E(\mathbf{x}_i, \mathbf{x}_j)} . \tag{5.7}$$

Yet another refinement strategy of the co-association matrix $\mathbf{S}$ is called the connected triple based similarity (CTS) and is a part of the link-based cluster ensemble algorithm (LCE) proposed by Iam-On et al. in 2010 [211]. There, the similarity matrix $\mathbf{S}$ is computed telling us how similar are clusters, but only for those that share some data points. Other similarities are zero. However, if cluster $\mathbf{c}_i$ is similar to cluster $\mathbf{c}_j$ and cluster $\mathbf{c}_j$ is similar to cluster $\mathbf{c}_k$, then we assume that also clusters $\mathbf{c}_i$ and $\mathbf{c}_k$ are similar to some degree even if there is no link between them.

So far, we briefly described the methods PAC, WEA and LCE that construct the similarity matrix $\mathbf{S}$ with more information about partitions compared to pure EAC. Thus, we consider them as the weighted evidence accumulation approaches. As we already said, another possibility to weight the partitions in the ensemble before merging them into consensus is by considering their quality or relevance and not only their objective properties. We call this approach the *partition relevance analysis* (PRA).

The motivation for using PRA comes from reasoning that not all the partitions from the ensemble should contribute equally to the consensus partition if they are not of the same quality. In the majority of cases where the cluster analysis is applied, we do not have any external or objective information of how the good-quality partition looks like. Thus, without ground-truth, we can only use the internal cluster validity indices (CVIs) for quality estimation, as we have already seen in Chapter 3. The first paper about PRA was written by Duarte et al. in 2005 [13], where the following methodology was proposed:

1. Generate the ensemble $\mathbf{P}$ of M partitions; $\mathbf{P} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_M\}$.
2. Validate each partition from $\mathbf{P}$ with selected CVI and obtain M values: $r_m = \mathrm{CVI}(\mathbf{C}_m)$, $\mathbf{r} = \{r_1, \dots, r_m, \dots, r_M\}$.
3. Normalize values of $\mathbf{r}$ to the interval $[0, 1]$. Mind different CVI types:
   - if CVI is max-like[3]: $R_m = \frac{r_m}{\max \mathbf{r}}$;

[3]The higher the value of the CVI, better the partition.

- if CVI is min-like[4]: $R_m = \frac{\min \mathbf{r}}{r_m}$.

4. Weight w_m of the partition $\mathbf{C_m}$ becomes the value of R_m. Compute the weighted similarity matrix $\mathbf{S}$ as:

$$\mathbf{S}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \lambda_{ij}^{\mathrm{EA}}(\mathbf{C_m}) \cdot w_m \,. \tag{5.8}$$

5. Employ an arbitrary clustering algorithm to detect K clusters from $\mathbf{S}$.

Evidently, the authors proposed the usage of one CVI at the time. Thus, they labelled this approach as single weighted evidence accumulation clustering (SWEAC). Duarte et al. experimented with 12 CVIs and they experimentally proved that using the PRA step leads to generally better results. However, none of the selected CVIs performed systematically better than others and it is hard to select the appropriate CVI without an expert knowledge. In the following study [14] the same authors proposed the joint weighted evidence accumulation clustering (JWEAC), where the weight w_m of the partition $\mathbf{C_m}$ is computed as arithmetic mean of all the I CVIs' values:

$$w_m = \frac{1}{I} \sum_{i=1}^{I} R_{mi} \,, \tag{5.9}$$

where R_{mi} is the normalized value of the i-th CVI on the partition $\mathbf{C}_m$. The published results are promising and suggest further investigation on using CVIs in the context of cluster ensembles. Later, a related study was carried out by Vega-Pons et al. in 2010 [10], where they investigated the benefits of the PRA step based on four CVIs before applying the kernel-based consensus function. Their work is not related with the evidence accumulation principle, but we are interested in two novel aggregation functions they introduced, which compute the weights from the CVIs' values. They considered two scenarios: a) Clustering with Lack of Knowledge (CLK) and b) Clustering with Background Knowledge (CBK). We refer to them in the following section.

The PRA step itself can be seen as an ensemble problem: each partition is evaluated by multiple judges and the final weights have to reflect the consolidation between

[4]The lower the value of the CVI, better the partition.

them. In the literature this approach is known as ensembles of clustering validity indices and is thoroughly assessed in the recent study of Vendramin et al. [225]. They included 28 CVIs and systematically evaluated all the combinations of 3 and 5 CVIs in the ensemble. The main question is, whether the combination of CVIs outperforms the single CVI. The consolidated values of CVIs are computed using the so called aggregation function. They applied four aggregation functions, one of them being the average like in Eq. (5.9). They concluded that blindly-built CVI ensembles are robust to the worst CVI in the ensemble, but do not outperform the best one in general.

In 2015, Jaskowiak et al. continued the research in the direction of guided selection of the CVIs that participate in the ensemble [49]. They employed the same 28 CVIs as in [225] and presented a strategy of selecting the CVIs in more informed and supervised way. As it is known from the ensemble-learning literature, the diversity or complementary and the accuracy or effectiveness are the key properties that influence the quality of the ensemble solution. The authors measured the diversity between CVIs by correlation coefficient between all the pairs on a huge amount of testing partitions. The accuracy of the individual CVI was assessed using the ground truth information about perfect clustering of a dataset: CVI's validates the partition and the same does the external validity index (ARI) with known ground-truth; the correlation between the CVI and external index is then used as a measure of effectiveness. Note, that we address this methodology of CVI evaluation in Chapter 3 as the *alternative-all* methodology. Jaskowiak et al. showed that their strategy of selecting the CVIs that are enough diverse and accurate, outperforms the blind selection presented in [225]. The CVI ensembles performed even better than the best single CVI in the ensemble. However, we have to highlight the fact, that this approach uses an external information about the desired or ground-truth partition of a certain dataset, which is usually not available to the data-miner in the real-world situation when facing unlabelled data. Yet, their study reveals that building ensembles of CVIs is beneficial and worth trying. Another interesting conclusion of the discussed study is that the normalization of the CVIs' values has to be carried out carefully, because the values of different CVIs are not necessarily following the same probability distribution. Their answer to this issue is the utilization of rank-based normalization, which we also include in our work. The rank-based normalization was also used in the research performed by Naldi et al. in 2013 [226], where they devised a few cluster ensemble selection strategies based on the PRA principle using six CVIs.

So far, we have walked through studies that encourage the weighting of the co-association matrix using various techniques, one of them being based on the committee or ensemble of internal cluster validity indices. Such an ensemble should be selected carefully to ensure enough diversity and accuracy. We have seen that three approaches had been proposed so far:

- no selection [10, 14, 226],
- random selection [225], and
- supervised selection based on the external validity measure [49].

Our work extends those by introducing the unsupervised selection of CVIs using feature-selection and feature-extraction algorithms. We propose an enhancement of PRA with the additional *reduction* step (PRAr) that reduces the number of CVIs while preserving the most informative ones. In the next section, we present our contribution more formally.

5.3 Partition relevance analysis with reduction step

First, let us formulate the original PRA approach. Basically, it consists of two steps: unification of CVIs' values and their aggregation into weights. The unification step ensures that CVIs' values become comparable among different CVIs by applying the normalization and transformation of the min-like CVIs into max-like. More formally, let $\mathbf{r}$ be a matrix of raw CVIs' values of size $M \times I$, where M is the number of partitions in the ensemble and I is the number of CVIs used. So, $r_{mi} = \text{CVI}_i(\mathbf{C_m})$. Let Γ be the unification function that transforms the matrix $\mathbf{r}$ into the matrix $\mathbf{R}$ of values on the same interval and with the same interpretation: large values means higher relevance. So, $R_{mi} = \Gamma_{\max}(r_{mi})$. We have already defined such a function proposed by Duarte et al. [13]. We call it the $\Gamma_{\max}$ unification function:

$$\Gamma_{\max}(r_{mi}) = \begin{cases} \frac{r_{mi}}{\max_{1\le j\le M} r_{ji}}, & \text{if CVI}_i \text{ is max-like,} \\ \frac{\min_{1\le j\le M} r_{ji}}{r_{mi}}, & \text{if CVI}_i \text{ is min-like.} \end{cases} \tag{5.10}$$

Moreover, here we define another three unification functions: Γ_{range}, Γ_{prob}, and Γ_{rank}. The unification with normalization on the interval $[0, 1]$ and mirroring the values of the min-like CVIs over the mid-point is called Γ_{range} and is defined as

$$\Gamma_{\text{range}}(r_{mi}) = \begin{cases} \frac{r_{mi} - \min_{1\le j\le M} r_{ji}}{\max_{1\le j\le M} r_{ji}}, & \text{if CVI}_i \text{ is max-like,} \\ 1 - \frac{r_{mi} - \min_{1\le j\le M} r_{ji}}{\max_{1\le j\le M} r_{ji}}, & \text{if CVI}_i \text{ is min-like.} \end{cases} \tag{5.11}$$

The function Γ_{prob} turns CVIs' values into probabilities. The values of min-like indices have to be mirrored over the mid-point before:

$$r^{\dagger}_{mi} = \max_{1\le j\le M} r_{ji} + \min_{1\le j\le M} r_{ji} - r_{mi}\,. \tag{5.12}$$

So, we define the Γ_{prob} as

$$\Gamma_{\text{prob}}(r_{mi}) = \begin{cases} \frac{r_{mi}}{\sum_{1\le j\le M} r_{ji}}, & \text{if CVI}_i \text{ is max-like,} \\ \frac{r^{\dagger}_{mi}}{\sum_{1\le j\le M} r^{\dagger}_{ji}}, & \text{if CVI}_i \text{ is min-like.} \end{cases} \tag{5.13}$$

As justified by Jaskowiak et al. [49], the rank-based normalization is free of any assumptions about the underlying CVI score's distribution. Thus, we also consider the function Γ_{rank} that unifies CVIs' values using their ranks. First, let us define the function `rank` that assigns the value M to the highest CVI value, the value $M - 1$ to the second largest and so on to the smallest CVI value, which gets the rank of 1. Ties are resolved using the average ranks. Now, the Γ_{rank} is formulated as follows

$$\Gamma_{\text{rank}}(r_{mi}) = \begin{cases} \frac{\mathtt{rank}(r_{mi})}{M}, & \text{if CVI}_i \text{ is max-like,} \\ \frac{1-\mathtt{rank}(r_{mi})}{M} + 1, & \text{if CVI}_i \text{ is min-like.} \end{cases} \tag{5.14}$$

When the CVIs' values are unified in the matrix $\mathbf{R}$, we can join them together into the weights $\mathbf{w}$ using the aggregation function Ω: $w_m = \Omega(\mathbf{R}_{m\bullet})$, where $\mathbf{R}_{m\bullet}$ is a m-th row of the matrix $\mathbf{R}$. In this thesis we limit ourselves to the following five aggregation functions: Ω_{mean} [13], Ω_{min}, Ω_{CBK} [10], Ω_{CLK} [10], and Ω_{RRA} [227]. They are defined as

$$\Omega_{\text{mean}}(\mathbf{R}_{m\bullet}) = \frac{1}{I}\sum_{i=1}^{I} R_{mi}\,, \tag{5.15}$$

$$\Omega_{\text{min}}(\mathbf{R}_{m\bullet}) = \min_{1\le i\le I} R_{mi}\,, \tag{5.16}$$

$$\Omega_{\mathrm{CBK}}(\mathbf{R}_{m\bullet}) = \sum_{i=1}^{I}\left(1 - \left|R_{mi} - \max_{1\le j\le M} R_{ji}\right|\right), \tag{5.17}$$

$$\Omega_{\mathrm{CLK}}(\mathbf{R}_{m\bullet}) = \sum_{i=1}^{I}\left[H(\mathbf{R}_{\bullet i})\left(1 - \left|R_{mi} - \frac{1}{M}\sum_{j=1}^{M} R_{ji}\right|\right)\right], \tag{5.18}$$

where $\mathbf{R}_{\bullet i}$ is the i-th column of the matrix $\mathbf{R}$ and $H(\mathbf{R}_{\bullet i}) = -\sum_{j=1}^{M} R_{ji}\log R_{ji}$ is the entropy of the i-th column.

The aggregation function Ω_{RRA} refers to the *Robust Rank Aggregation* algorithm proposed by Kolde et al. in 2012. It is one of the many rank aggregation algorithms and was proposed especially for the cases when there are noise and outliers present among the rankings being aggregated. The effectiveness of the RRA algorithm was demonstrated in the field of genomics for the integration of ordered gene-lists into the final one. The RRA is founded on the statistical analysis of rankings – it tries to find the partitions in the ensemble that perform consistently better than expected under null hypothesis of uncorrelated rankings. Then it produces the final ranking of the partitions based on their significance scores. We assume the unification function Γ_{range} is used in a combination with Ω_{CBK}, Γ_{prob} with Ω_{CLK}, and Γ_{rank} with Ω_{RRA}.

Now, let's dig into our proposal – the partition relevance analysis with reduction step (PRAr). The main idea is to build a system that can select the relevant CVIs among the supplied list automatically without external information, expert knowledge or the intervention of the user. In a real-world situation we usually do not know, which CVI is the best for our data or how many of them we need in the ensemble for the PRA. To tackle this challenge, we propose to validate ensemble partitions with multiple CVIs (in our experiments we used 34 of them). After the *unification* of obtained values we add the *reduction step* that reduces the number of CVIs, i.e. the columns of the matrix **R**, in order to filter out redundancy and noise before the *aggregation* function steps in action. Well, the approach itself is not new, far from it – this is simply a dimensionality reduction phase, better known as feature selection or feature extraction. Feature selection methods output a subset of original features, whereas feature extraction methods construct new features based on the original ones. The ultimate goal of such methods is to eliminate redundant, noisy, or irrelevant features to alleviate learning of supervised or unsupervised algorithms. For the overview of the field and survey of the state-of-art algorithms for feature selection/extraction, refer to [228–231].

One important finding of the already mentioned study by Jaskowiak et al. [49] is that among dozens of published CVIs there is a lot of redundancy, meaning that many CVIs perform more or less the same on certain data. So, if we build our CVI ensemble of members that are not diverse enough, the principles of ensemble-learning do not work. Hence, the reduction of similar CVIs is essential. In our study we experimented with three feature selection methods that are suitable for unsupervised learning: Laplacian Score (LS) [232], spectral feature selection (Spec) [233], and feature selection using k-medoids[5] (FSKM) [234]. In addition, we employ also two feature extraction methods: probabilistic principal component analysis (PPCA) [235, 236] and feature extraction using k-means (FEKM) [228].

Let us define $\Pi(\mathbf{R})$ to be a reduction function that maps matrix $\mathbf{R}$ with I columns to matrix $\hat{\mathbf{R}}$ with $\hat{I}$ columns, where $\hat{I} \leq I$. Besides the functions Π_{LS}, Π_{Spec}, Π_{FSKM}, Π_{PPCA}, and Π_{FEKM} we experiment also with the function Π_{none} that leaves matrix $\mathbf{R}$ intact, i.e. $\hat{I} = I$.

Common to all the listed reduction functions is their dependency on the user-defined number of selected/extracted features. This means, that we have to specify $\hat{I}$ in advance. In order to automatize the whole procedure as much as possible, we propose to employ an intrinsic-dimensionality estimator. Let $\mathbf{X}$ be a dataset with D-dimensional data points – its intrinsic dimensionality is defined "as the minimum number of parameters needed to represent the data without information loss" [237]. The methods for intrinsic dimension estimation are commonly used in a combination with data reduction methods [238]. A comprehensive survey of the state-of-art estimators was written recently by Camastra and Staiano [239]. Following their review we chose a method DANCo[6], which stands for dimensionality from angle and norm concentration proposed by Ceruti et al. in 2014 [237]. Camastra and Staiano performed a comparison of the estimators and DANCo proved to be accurate and robust to high dimensionality.

The main idea of the DANCo algorithm is to compute the k-nearest neighbours (kNN) graph on the dataset and to measure two statistics based on the graph: the estimated probability density function (concentration) of the normalized distances between neighbours and the concentration of the angles between all possible pairs of

[5]As a dissimilarity measure between features we implemented the Euclidean distance, whereas Jiang et al. used a mutual information.

[6]We used the FastDANCo implementation that is a faster variant of DANCo with comparable accuracy.

neighbouring data points. These two statistics are supposed to be dependent on the intrinsic dimensionality of the dataset. The kNN method causes a systematic bias that overestimates or underestimates the true intrinsic dimensionality using the concentration of the angles or normalized distances, respectively. To provide a correction of the bias, DANCo compares the joint probability density functions related to angles and distances estimated on the dataset in question, with those estimated on synthetic datasets of known intrinsic dimensionality using the Kullback-Leibler divergence. The estimated intrinsic dimensionality of the dataset is the one that minimizes the Kullback-Leibler divergence. A synthetic dataset is generated for every possible intrinsic dimensionality $d \in \{1, \dots, D\}$ as unit d-dimensional hypersphere of N uniformly distributed data points.

In the following section we describe our experimental study, where we construct cluster ensembles of $M = 20$ partitions that are evaluated by 34 CVIs, thus we get the matrix $\mathbf{R}$ of size 20×34. We interpret $\mathbf{R}$ as a new dataset with 20 data objects of 34 dimensions, which is regarded as a high-dimensional problem.

To sum up: in this section we propose an enhancement of the partition relevance analysis with the additional reduction step – PRAr. A complete pipeline of PRAr is sketched as

$$\mathbf{P} \xrightarrow{\text{CVIs}} \mathbf{r} \xrightarrow{\Gamma} \mathbf{R} \xrightarrow{\Pi} \hat{\mathbf{R}} \xrightarrow{\Omega} \mathbf{w}\,. \tag{5.19}$$

We used the weights $\mathbf{w}$ obtained in the PRAr to refine the co-association matrix $\mathbf{S}$ as defined in Eq. (5.8). Then we plugged it into the core of three consensus functions[7], namely EAC, CSPA, and DICLENS. Thus, we get three new consensus functions based on the PRAr: EAC-W, CSPA-W, and DICLENS-W that are compared with nine others in the next section.

5.4 *Experiments*

Here we describe our experimental work through which we evaluate our contributions on the multiple levels:

- comparison of single-clustering algorithms including the proposed gSOM (see Chapter 4);

[7]We limit ourselves on the selected three methods, but the same modification can be done to every consensus function that is based on the evidence accumulation principle.

- comparison of cluster ensemble methods using various ensemble- generation settings and consensus functions; we experimented with enabled PRA/PRAr step and without it;
- comparison of single-clustering with the cluster ensemble approaches.

All the experiments were conducted on the same datasets families as for comparison of two-level clustering methods in Chapter 4: the synthetic `Complex2D` (see Tab. 4.2), the `GENE` (see Tab. 3.1) and the `REAL` datasets (see Tab. 3.2). For the external validation of the partitions produced by single-clustering algorithms as well as ensemble methods we employ three eCVIs, namely ARI, AMI and BCA. For their definitions, refer to Section 3.6.3. All the results are presented using the ARI measure, while we use AMI and BCA only for the most important ones for the sake of completeness. If not otherwise stated, we aggregate scores using the average over 30 independent runs.

5.4.1 *Parameters setting*

Remember that the cluster ensemble pipeline consists of three steps: ensemble generation, the partition relevance step, and the consensus step. In the following, we go through these steps one-by-one describing the implementation details.

Ensemble generation

To generate data partitions we used six single-clustering algorithms. Three of them are hierarchical by their nature and deterministic: average-linkage (AL), complete-linkage (CL), and single-linkage (SL) [50]. Other three are stochastic: k-means (KM) [102], spectral algorithm with local scaling (Sp) [103], and gravitational clustering of the self-organizing map (gSOM); the latter two have been already discussed in Chapter 4. Note, that there are two variants of Sp and gSOM algorithms: the first that automatically determines the number of clusters and the second that allows user to specify it. As the AL, CL, SL, and KM methods need the number of clusters K to be passed as a parameter, we use the second variant of Sp and gSOM. We chose those six single-clustering algorithms for the following reasons. First, the AL, CL, SL, and KM algorithms are the representatives of simple and frequently used clustering algorithms that are implemented in many open-source and commercial software packages (MATLAB, Mathematica, Octave, Orange, R, SAS, SPSS, Stata, Weka, etc.); the Sp algorithm is included due to its high accuracy and ability to discover clusters of non-spherical shape. Second, we consider only the algorithms that produce crisp partitions[8], so we do not address fuzzy or probabilistic approaches. Finally, we limit ourselves to the six algorithms due to a high computational cost of the experimental set-up.

The hierarchical algorithms AL, CL, and SL come from the family of the agglomerative clustering – it is a bottom-up approach, where each data point is a cluster at the beginning. In the each step two closest clusters are merged together and this is repeated until all clusters are merged into a single cluster that contains all the data points. The distance between a pair of clusters $d(\mathbf{c}_i, \mathbf{c}_j)$ is defined as the average (AL), the maximum (CL), or the minimum (SL) Euclidean distance between two data points, where one data point belongs to the cluster $\mathbf{c}_i$ and one to the cluster $\mathbf{c}_j$:

$$d(\mathbf{c}_i, \mathbf{c}_j) = \begin{cases} \frac{1}{|\mathbf{c}_i| \cdot |\mathbf{c}_j|} \sum_{\mathbf{x}_i \in \mathbf{c}_i} \sum_{\mathbf{x}_j \in \mathbf{c}_j} d_E(\mathbf{x}_i, \mathbf{x}_j), & \text{average-linkage;} \\ \max \left\{ d_E(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \in \mathbf{c}_i,\ \mathbf{x}_j \in \mathbf{c}_j \right\}, & \text{complete-linkage;} \\ \min \left\{ d_E(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \in \mathbf{c}_i,\ \mathbf{x}_j \in \mathbf{c}_j \right\}, & \text{single-linkage.} \end{cases} \tag{5.20}$$

We set the number of iterations of the KM to 100. The parameter `kNN` of the Sp algorithm, which controls the local scaling of the σ value was optimized over the datasets using the same three evaluation protocols as for gSOM in Chapter 4: the eCVI, CV, and CVI protocol. We tested a set of values $[1, 2, 3, 5, 7, 9, 11, 15]$ when the data is clustered into K_T clusters, where K_T equals the number given as ground-truth. Exactly the same was done for the gSOM algorithm – the parameters and their sets of values are presented in detail in Section 4.5.4. We refer to these optimal values as the best parameters' configuration with respect to the mentioned evaluation protocols. For ensemble generation we consider the CVI protocol, where we optimize the parameters of Sp and gSOM using the same internal validity indices as in Chapter 4, i.e. DNs for the `Complex2D`, and CH for the `GENE` and `REAL` datasets. However, for the purposes of the comparison between single-clustering algorithms in Section 5.4.2, we report the results of the eCVI and CV protocol as well.

To ensure diversity in the ensemble, we applied three data subsampling strategies and three strategies for the selection of the parameter K. Datasets were subsampled by their features using 100% (no subsampling), 25–50%, and 75–85% of randomly selected features; as the `Complex2D` datasets have only two features, we did not apply

[8]Each data point belong to one cluster only.

subsampling on them. The number of clusters K in each partition of ensemble was selected using the following three schemes:

- K_T: number of clusters equals the number of clusters in the ground truth partition;
- $\rightarrow \sqrt{N}$: number of clusters is selected from the interval $\left[2, \lceil \sqrt{N} \rceil\right]$ at random, where N is the number of data points in the dataset;
- $\sqrt{N} \rightarrow$: number of clusters is selected randomly from the interval $\left[\lceil \sqrt{N} \rceil, \lceil \sqrt{N} \rceil + 20\right]$. Upper limit is $N/2$.

For each combination of feature subsampling and selection of K scheme a single-clustering algorithm produced an ensemble of 20 partitions, i.e. the size of ensemble is $M = 20$. The partitions created using all the data features (100%) and containing K_T clusters were selected for the comparison between single-clustering methods, which is discussed in Section 5.4.2.

PRAr

In the partition relevance analysis with reduction step we experimented with different configurations of functions for unification, reduction and aggregation of the cluster validity indices' values, as described in Section 5.3. Every partition from the ensemble was validated by 34 CVIs that are presented in Chapter 3 along with the proposed DNs index. Thus, the matrix $\mathbf{r}$ has 20×34 entries. The four unification functions are used to bring the values of CVIs together: Γ_{max}, Γ_{range}, Γ_{prob}, and Γ_{rank}. After the unification we considered two scenarios: a) we reduced the matrix $\mathbf{R}$ using one out of five reduction functions Π_{LS}, Π_{Spec}, Π_{FSKM}, Π_{FEKM}, or Π_{PPCA} or b) we skipped the reduction, which equals to the usage of the function Π_{none}; this also means that in this case we are considering the PRA and in the case of a) the PRAr approach. In the last step we employed one of five different aggregation functions: Ω_{mean}, Ω_{min}, Ω_{CBK}, Ω_{CLK}, or Ω_{RRA}.

Some reduction functions have parameters to set up. For Π_{LS} we set the number k for the k-nearest neighbours search to 5 as suggested by the authors [232]. The k-medoids and the k-means algorithms ran 50 times and the solution that minimizes the sum of distances to cluster's prototypes is considered for the Π_{FSKM} and Π_{FEKM}, respectively. We limit the expectation-maximization algorithm integrated in the Π_{PPCA}

to 200 iterations. As suggested by the authors of the DANCo estimator, we set its parameter k for the k-nearest neighbours search to 10.

We considered all the possible combinations between unification, reduction and aggregation functions, where we acknowledged the following constraints: Ω_{CBK} requires Γ_{range}, Ω_{CLK} requires Γ_{prob}, and Ω_{RRA} is constrained by unification function Γ_{rank}. In total we have 66 configurations of the PRAr.

Consensus functions

Each ensemble of partitions was consolidated by 12 consensus functions we described in the previous two sections. We list them in the chronological order of their proposal:

- evidence accumulation clustering (EAC) [198],
- cluster-based similarity partitioning algorithm (CSPA) [192],
- hypergraph partitioning algorithm (HGPA) [192],
- meta-clustering algorithm (MCLA) [192],
- joint weighted evidence accumulation clustering (JWEAC) [14],
- probability accumulation clustering (PAC) [221],
- weighted evidence accumulation using co-association significance (WEA) [224],
- link-based cluster ensemble (LCE) [211],
- divisive clustering ensemble with automatic cluster number (DICLENS) [220],
- EAC using PRAr (EAC-W),
- CSPA using PRAr (CSPA-W),
- DICLENS using PRAr (DICLENS-W).

All the mentioned consensus functions require the desired number of clusters K to be given as parameter, except for the DICLENS algorithm. For the sake of fair comparison, we modified it to return K clusters when possible. Consensus functions EAC, JWEAC, PAC, WEA, LCE, and EAC-W compute the consensus partition by applying a single-clustering algorithm to the co-association matrix. We decided to use the single-linkage (SL) algorithm. The only algorithm that require an additional parameter is LCE, where we have to define the so called decay factor. It is a confidence level of accepting two non-identical clusters as being similar – the value of 0.9 is used as in [211] and [240].

The whole cluster ensemble procedure, i.e. ensemble generation → PRAr → consensus function, was repeated 30 times and the average of ARI, AMI, or BCA is regarded as the performance score.

5.4.2 Comparison of single-clustering algorithms

Our first analysis is that of single-clustering algorithms performance with no datasets' features subsampling and with the ground truth number of clusters K_T known to clusterers. This comparison between algorithms can be seen as an extension to that made in Chapter 4 (Section 4.5), where we compared the gSOM algorithm to several others based on the self-organizing map. Here, we compare gSOM to more *conventional* algorithms. We followed the same procedure of comparing algorithms on multiple datasets as in Chapters 4 and 3. We computed the algorithms' rank for every dataset using the average performance score over 30 repetitions using the optimal parameters' configuration considering the eCVI, CV, and CVI protocols. Then we computed the average of ranks over datasets and tested a null hypothesis that ranks are equal with the Friedman non-parametric test [107]. If the null hypothesis was rejected, a pairwise comparison was made between clustering algorithms using Bergmann-Hommel's post hoc procedure for multiple hypothesis testing [183] with 95% confidence level, i.e. $\alpha = 0.05$.

Fig. 5.1 demonstrates the results of statistical comparison on the `Complex2D` datasets. The Sp and SL algorithms perform very well with perfect scores for almost all the synthetic datasets regardless of the protocol. Statistical testing reveals no significant differences in performance between gSOM and best-ranked Sp algorithm, except for the CVI protocol. However, the scores' distribution differs noticeably with gSOM having wider spread and lower average and median.

The gSOM method has significantly higher expressiveness, measured by the eCVI protocol, than other algorithms on the `GENE` datasets as we can see in Fig. 5.2. When optimizing its parameters using the CV protocol, it is still significantly better than SL, but its mean score drops substantially. Somewhat better is with the CVI protocol,

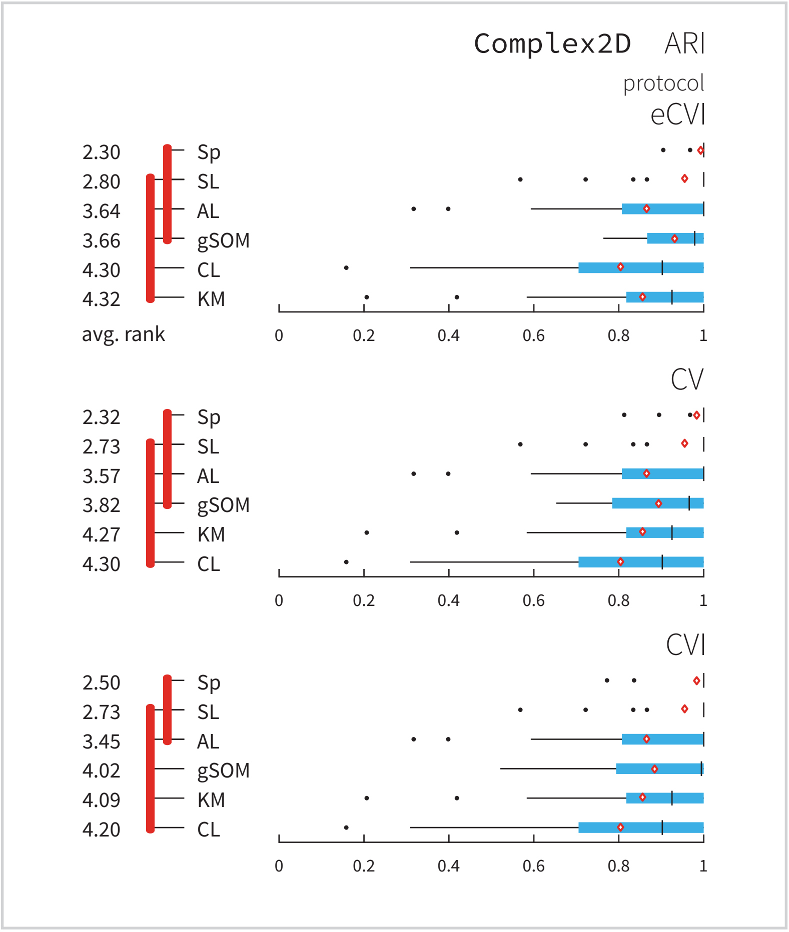


Figure 5.1

Average ranks of algorithms based on the ARI score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

where gSOM is placed into the group of methods that are better than hierarchical algorithms CL, AL, and SL. Considering the results on `REAL` datasets in Fig. 5.3, gSOM is significantly better than all the hierarchical algorithms in comparison.

We show the results based on the AMI and BCA scores in Appendix B. The methods' ranks are consistent with ARI with some minor changes as you can see from Figs. B.1,

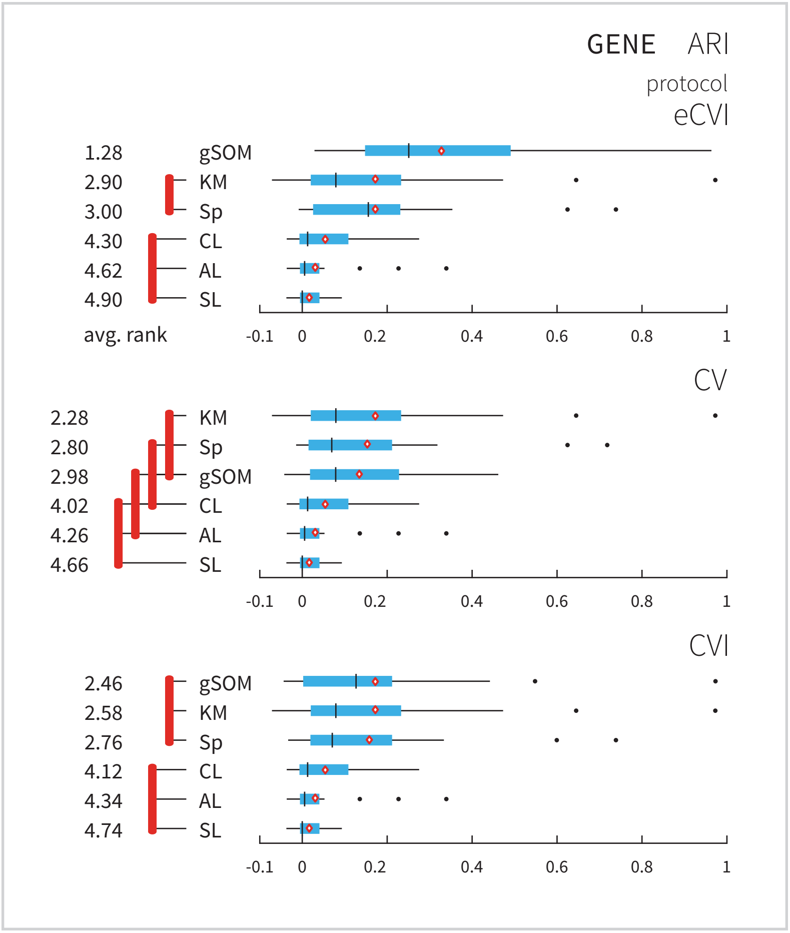


Figure 5.2

Average ranks of algorithms based on the ARI score across all the `GENE` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

B.2, B.3, B.4, B.5, and B.6. Those changes include: gSOM is not being significantly worse than winning Sp on `Complex2D` datasets with CVI protocol; considering `GENE` datasets, BCA measure, and eCVI protocol gSOM is significantly better only than CL, AL, and SL, but not Sp and KM; finally, regarding `REAL` datasets with AMI and BCA measures, gSOM is significantly better than hierarchical algorithms, except for the CV

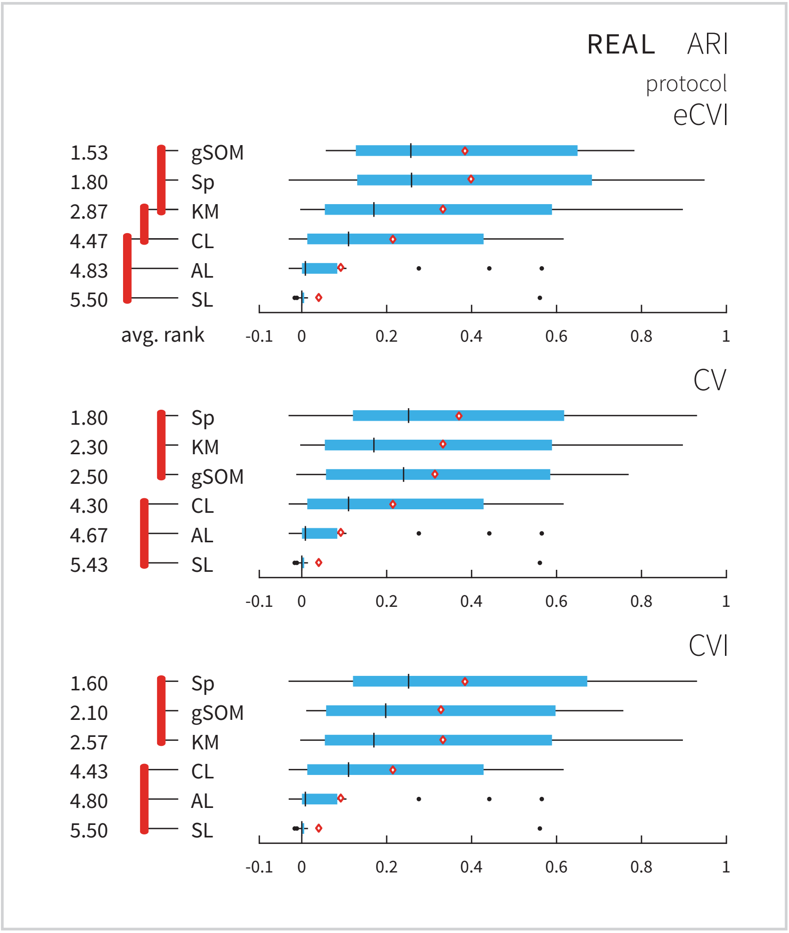


Figure 5.3

Average ranks of algorithms based on the ARI score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

protocol, where it is better than AL and SL, but not CL.

Although we compare single-clustering algorithms primarily by their accuracy, we should briefly consider their time complexity as well. With respect to the number of data points N, KM and gSOM have time complexity of $O(N)$, whereas AL, CL, SL, and Sp are not linear-time algorithms – they find the clustering solution in $O(N^2 logN)$

time.

5.4.3 Comparison of cluster ensemble methods

We performed a similar comparison between consensus functions as with the single-clustering algorithms. Here, each consensus function has computed the consensus partition for all the possible settings of the ensemble-generation process: 3 feature subsampling schemes, 6 single-clustering algorithms, and 3 strategies for selecting the number of clusters in the ensemble's partitions, in total of 54 different settings. Well, this is not the case with the `Complex2D` datasets for they were not subsampled due to their low-dimensionality. For the comparison, the average performance score over 30 repetitions is used considering the best ensemble-generation setting and the average of all settings. We analyse these settings thoroughly in the following subsection.

When considering the configurations of the consensus functions with the PRAr step enabled, i.e. EAC-W, CSPA-W, and DICLENS-W, we have to go one level deeper. For each of 54 ensemble-generation settings we applied 66 PRAr configurations and evaluated the results. Our goal was to find the PRAr configuration that performs the best on average across all the datasets of certain type. So, we first found the ensemble-generation setting that maximizes the performance measure for each of PRAr configuration on a dataset. Then the average across datasets was taken and the configuration that maximized this average score was considered in the following comparisons. Tab. 5.1 displays the PRAr configurations selected in this way. It is possible, that multiple configurations achieved the same best score.

Table 5.1

The best PRAr configurations on average across datasets.

cons. fun.	`Complex2D`	`GENE`	`REAL`
CSPA-W	$\Gamma_{\text{range}} - \Pi_{\text{LS}} - \Omega_{\text{min}}$	$\Gamma_{\text{prob}} - \Pi_{\text{none}} - \Omega_{\text{min}}$	$\Gamma_{\text{max}} - \Pi_{\text{PPCA}} - \Omega_{\text{mean}}$
DICLENS-W	all	$\Gamma_{\text{prob}} - \Pi_{\text{PPCA}} - \Omega_{\text{CLK}}$	$\Gamma_{\text{prob}} - \Pi_{\text{Spec}} - \Omega_{\text{min}}$
EAC-W	almost all[a]	$\Gamma_{\text{range}} - \Pi_{\text{none}} - \Omega_{\text{min}}$	$\Gamma_{\text{prob}} - \Pi_{\text{LS}} - \Omega_{\text{min}}$

[a]Exceptions: $\Gamma_{\text{range}} - \Pi_{\text{none}} - \Omega_{\text{min}}$, $\Gamma_{\text{range}} - \Pi_{\text{LS}} - \Omega_{\text{mean}}$, $\Gamma_{\text{range}} - \Pi_{\text{LS}} - \Omega_{\text{min}}$, $\Gamma_{\text{range}} - \Pi_{\text{Spec}} - \Omega_{\text{mean}}$, $\Gamma_{\text{range}} - \Pi_{\text{Spec}} - \Omega_{\text{min}}$.

Looking at top panel of Fig. 5.4 we can say it is hard to say anything about the best

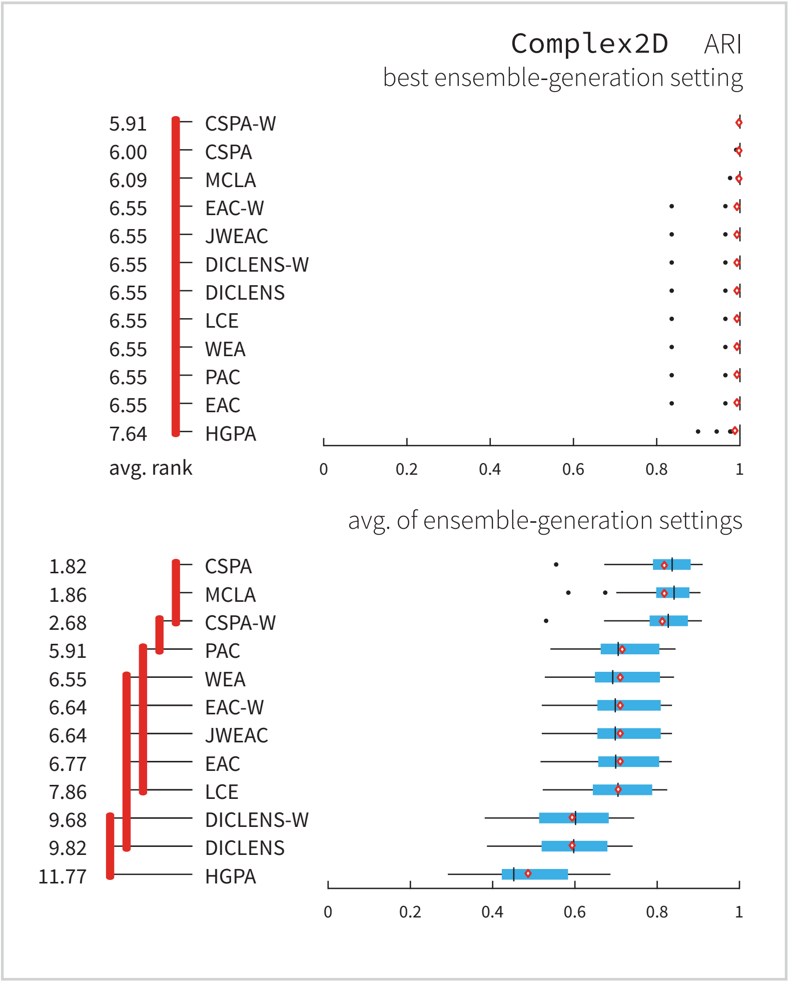


Figure 5.4

Average ranks of consensus functions based on the ARI score across all the `Complex2D` datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

or the worst algorithms for the `Complex2D` data. The Friedman's test failed to reject null hypothesis of equal ranks, which means the average ranks are not significantly different among the competitors. There is hardly any difference between them. Why this happened? The answer is in the scores distribution, where we see nearly perfect results

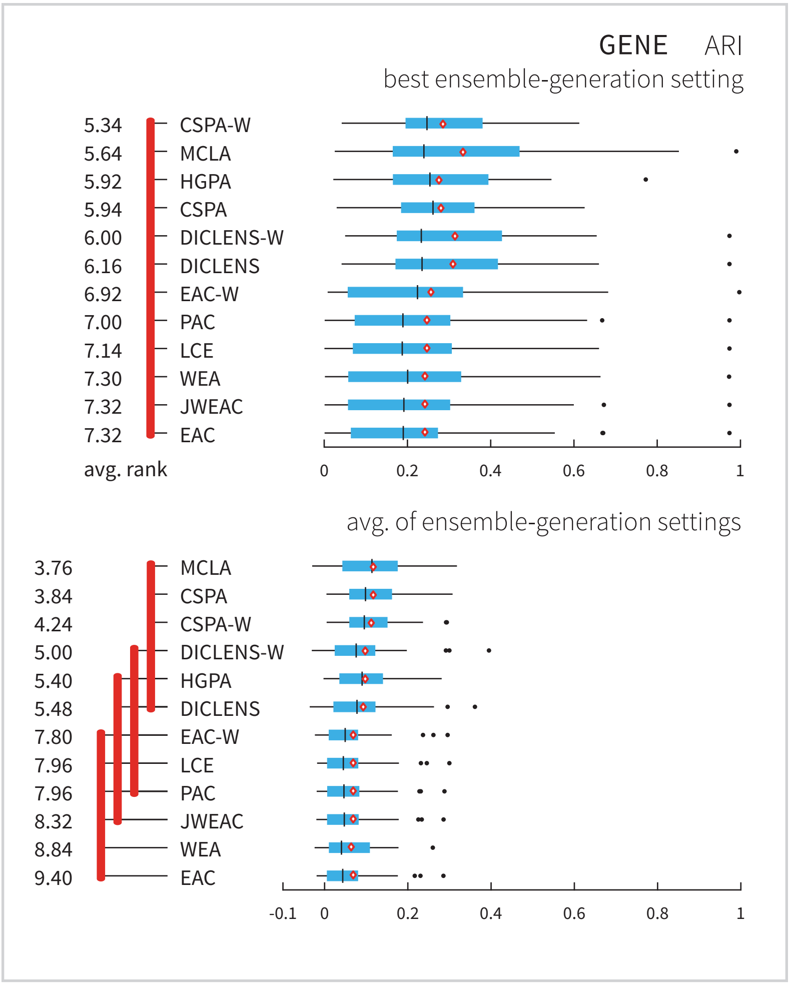


Figure 5.5

Average ranks of consensus functions based on the ARI score across all the GENE datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

regardless of the consensus function. Actually, this is good news for it indicates the superiority of the cluster ensembles performance. Also, the methods with enabled PRAr step (EAC-W, CSPA-W, DICLENS-W) perform as good as their simpler counterparts EAC, CSPA, and DICLENS. Again, not much difference when switching from ARI

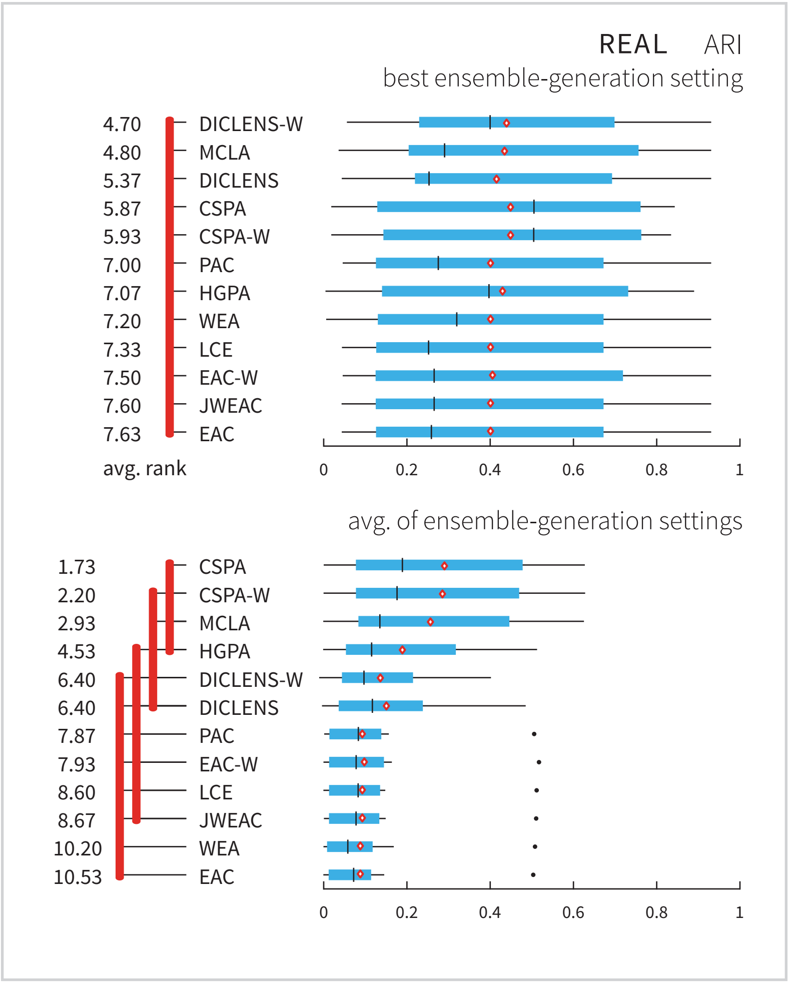


Figure 5.6

Average ranks of consensus functions based on the ARI score across all the REAL datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

to AMI and BCA measures as can be seen from Figs. B.7 and B.10 in the appendix. The averages of all ensemble-generation settings result in quite lower scores as can be seen from bottom panel of the figures. From this observation we can conclude that not all of the settings for ensemble generation are suitable for certain consensus function. There are no common guidelines for choosing the ensemble-generation settings that work well for all the consensus functions in general. For this reason, we primarily report the results based on the best setting. Hence, we have to consider these results similar as those based on the eCVI protocol in single-clustering algorithms comparison – as the upper bound of an algorithm's performance.

Let us move on to the GENE datasets. Again, the average ranks among consensus functions are not different significantly as we see from the top of Fig. 5.5. However, in the ranked lists for GENE datasets, the algorithms with incorporated PRAr step are often placed above their versions without the PRAr – this is always true for DICLENS-W and EAC-W, even when considering the AMI and BCA scores in Figs. B.8 and B.11. In the latter we notice somewhat greater degree of discrimination among algorithms. We suppose this is due to the fact that BCA is quite different measure from ARI and AMI. Indeed, further investigation and the comparison between external validity measures is needed, but is rather out of this thesis' scope. We observe a drop in performance when all the ensemble-generation settings are taken to compute the average behaviour of the consensus functions. The ordering of the algorithms is preserved in general with MCLA, CSPA, and CSPA-W being better than EAC-W, LCE, PAC, JWEAC, WEA, and EAC.

Our suspicion that the performance of consensus functions in comparison is more similar than not proves to be true also in the case of REAL datasets. The statistical test fails to distinguish between average ranks as is evident from Fig. 5.6. The PRAr seems to enhance the performance of DICLENS and EAC at least by little with the best ensemble-generation setting. This is not true for DICLENS when considering the average of all settings. Additional results using the AMI and BCA measures can be found in Figs. B.9 and B.12.

We can conclude that there is no significant difference in performance between the algorithms with PRAr step and their variants without PRAr. However, the consensus functions with PRAr tends to have a bit lower, therefore better ranks, which is especially true for EAC-W.

Analysis of cluster ensemble configurations

In our experiments there are many decision levels from the dataset at the beginning to the consensus partition at the end. Without considering the PRAr step, there are four of them and we refer to them in this context as to an *experimental configuration*. It is a quadruple of a form:

$$\Big([\% \text{ of features}], [\text{clustering algorithm}], [\text{selection of } K], [\text{consensus function}]\Big)\,. \tag{5.21}$$

For example, (25–50%, KM ,$\sqrt{N}$ $\rightarrow$, EAC) is one of them. We have $1 \cdot 6 \cdot 3 \cdot 12 = 216$ configurations for the `Complex2D` datasets and $3{\cdot}6{\cdot}3{\cdot}12 = 648$ configurations for `GENE` and `REAL` datasets. Each configuration achieved a score measured as average ARI over 30 repetitions for each dataset. Now, we are interested in the probability that a certain element of the configuration quadruple is a part of the winning configuration, i.e. the one which maximizes the average ARI score. For instance, what is the probability that we get best performing configuration, if we select KM clustering algorithms for the ensemble generation? In this way we assess the influence of the four experiment's parameters on the performance of the final solution.

The win probabilities $\boldsymbol{\beta}$ of m competing methods on n datasets were estimated using a simple Bayesian model [241], where the outcome, i.e. winner indicator vector, $\boldsymbol{y}_i, i = 1, \dots, n$ is assumed to follow a categorical distribution with probabilities $\boldsymbol{\beta}$, with a non-informative Dirichlet prior on $\boldsymbol{\beta}$, set to $1/2$ as suggested by Jeffreys [242]:

$$\begin{aligned} \boldsymbol{y}_i &\sim \text{Categorical}(\boldsymbol{\beta}),\ i = 1, \dots, n\,, \\ \boldsymbol{\beta} &\sim \text{Diriclet}\left(\frac{1}{2} \cdot \mathbf{1}_m\right)\,. \end{aligned} \tag{5.22}$$

Cases where two or more competing methods were tied for first place were treated as incomplete information – in such cases every competing method tied for first place is assumed to have an equal chance of being the actual winner and the likelihood is updated accordingly. The model was fit using the Bayesian inference tool `Stan` [243] (4000 samples, 1000 sample warm-up, all the standard errors of Markov-chain Monte Carlo procedure < 0.01). `Stan` code for the model is included in the Appendix section B.3.

In Fig. 5.7 we present results on `Complex2D` datasets. On the left-hand side panel we depict the win probability of certain consensus function. Mean probabilities are in favour for the LCE and CSPA-W functions, but the overlapping confidence intervals suggest we are not very certain about the difference between different consensus functions. These results are in accordance with the pairwise comparison with average

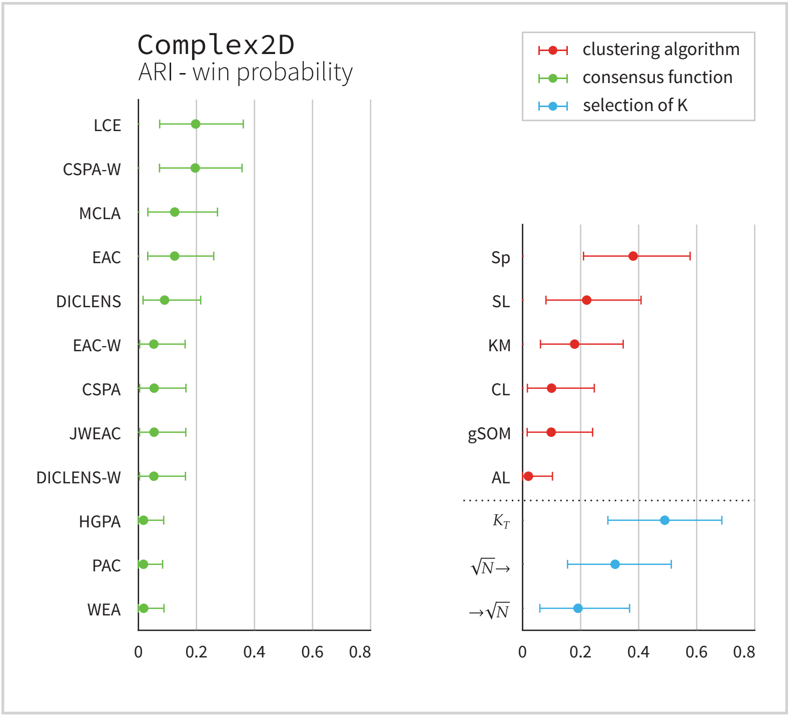


Figure 5.7

Estimated probability that a certain method is a member of the winning configuration that maximizes the average ARI score across 30 repetitions on `Complex2D` datasets. Confidence intervals of 95% are considered ($\alpha = 0.05$). Methods are sorted by the lower bound of their confidence intervals.

ranks presented in the previous section for the best ensemble-generation setting. On the right side we see that the experimental configuration has the highest probability of win if we select Sp as a clustering method. But, we are not so certain about it as indicated by a wide confidence interval that spans from around 20% to 58%. It also overlaps substantially with the SL method, so we cannot say the Sp is always the best choice for ensemble generation. However, we are quite sure, that in worst-case (at the lower bound) we should get better results than with other methods when we deal with a dataset from the same distribution as the `Complex2D` datasets. Furthermore, there are similar conclusions when choosing different strategies for the selection of K. Using K_T provided as ground-truth seems to be better choice.

The results on the `GENE` and the `REAL` datasets are presented in Figs. 5.8 and 5.9, respectively. They both suggest that CSPA algorithm is better choice than other con-

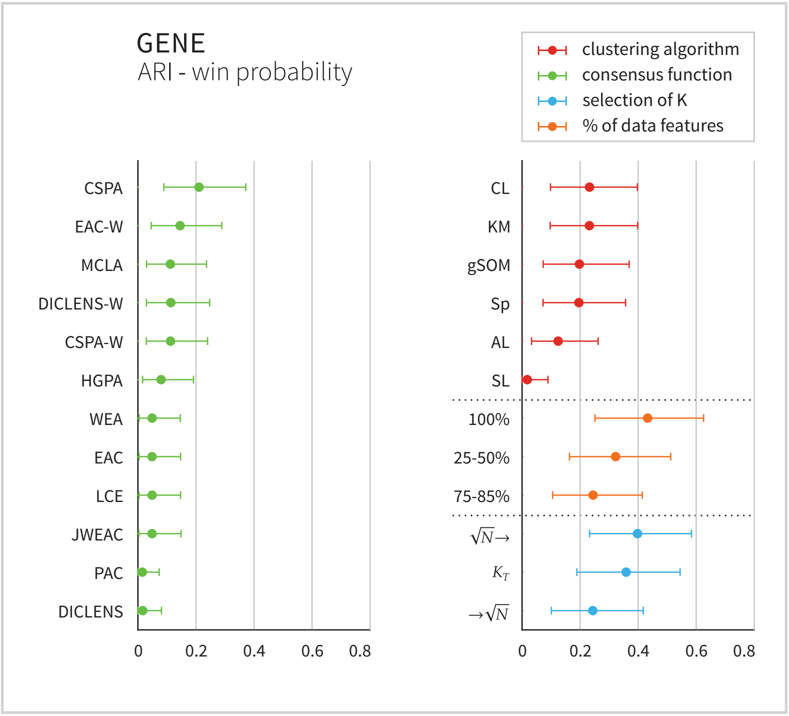


Figure 5.8

Estimated probability that a certain method is a member of the winning configuration that maximizes the average ARI score across 30 repetitions on GENE datasets. Confidence intervals of 95% are considered ($\alpha = 0.05$). Methods are sorted by the lower bound of their confidence intervals.

sensus functions. The ordering of functions differs somewhat from that of the pairwise comparison, yet the wide confidence intervals suggest that methods are quite similar in their performance. Other findings on GENE datasets include that in most of the cases the KM or CL clusterers, all the available data features, and the number of clusters bigger than $\sqrt{N}$ in the ensemble partitions is preferable. Be aware, that we can not assume that a combination of the already highlighted methods would also give the high probability of win. Regarding the REAL datasets, similar conclusion about feature subsampling and the number of clusters can be drawn. Here, the Sp and gSOM single-clustering algorithms have high win probability.

Yet similar but less powerful, way of analysing the influence of experimental configuration is to break down results by consensus functions and investigate, which configurations are performing the best. For each consensus function we search for the best

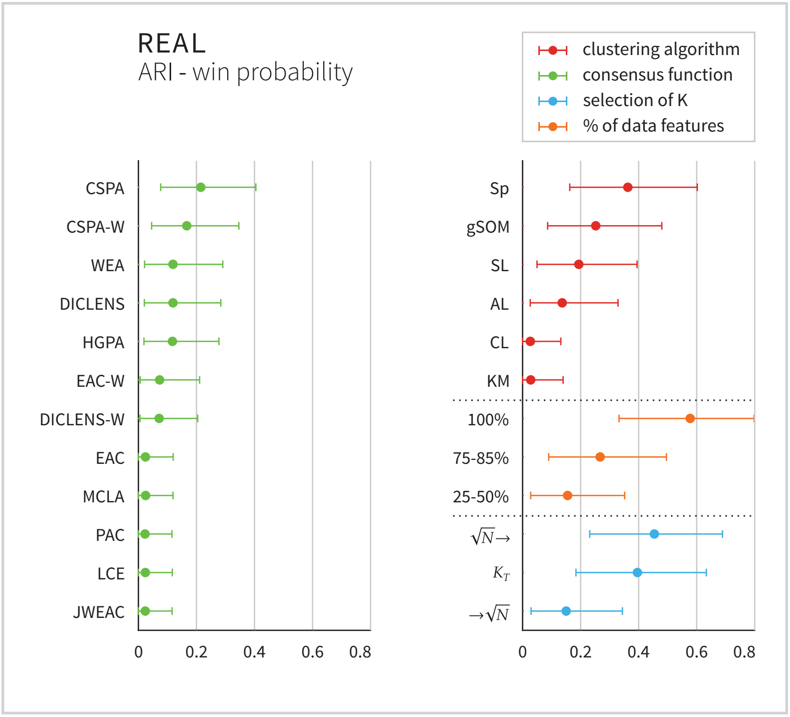


Figure 5.9

Estimated probability that a certain method is a member of the winning configuration that maximizes the average ARI score across 30 repetitions on REAL datasets. Confidence intervals of 95% are considered ($\alpha = 0.05$). Methods are sorted by the lower bound of their confidence intervals.

configuration of feature subsampling mode, cluster algorithm, and the selection strategy of K. Then we count over datasets how many times a particular option is included in the best configuration. Hence, we are examining, if there are differences in consensus function's preferences about the ensemble-generation setting. In Fig. 5.10 we show the number of times an option participates in the best configuration measured by the average of the ARI score across 30 repetitions on the `Complex2D` data. Please note that multiple configurations can achieve the best score, hence the counts can exceed the number of datasets. This is evidently the case here as the perfect results are achieved on the majority of `Complex2D` datasets with more than one configuration. The GENE and REAL datasets are considered in Figs. 5.11 and 5.12. In general, these results agree with those obtained from Bayesian modelling and offer us another insight into the consensus functions' behaviour over different settings.

Analysis of PRAr configurations

In Tab. 5.1 we collected the best PRAr configurations on average over the datasets. The results guide us to conclusion that the reduction step is beneficial as the majority of combinations include one of five reduction functions different from *none*.

We can try to give an answer to the question, which PRAr configuration to use also with Fig. 5.13. Now, we consider the overall best PRAr configurations. So, for each dataset we find the best ensemble-generation setting and the best PRAr configuration. Then we count the number of datasets, where certain option is a member of the winning combination. We display the results for each consensus function that is based on the PRAr enhancement.

As expected on `Complex2D`, there are no differences among configurations. Almost all of them achieved the perfect score. More versatile is the situation with `GENE` datasets. We see that Γ_{prob} is preferable by CSPA-W and DICLENS-W, and Γ_{range} by EAC-W. Considering the reduction functions, the configurations with Π_{PPCA} win more often in combination with the DICLENS-W algorithm. The EAC-W prefers Π_{none}, Π_{PPCA}, and Π_{FEKM}. The preferences of CSPA-W are not so distinguishable, all the reduction functions are quite frequently represented in the best configuration with Π_{FEKM} having a small lead. The differences between CSPA-W, DICLENS-W and EAC-W are the most obvious when it comes to aggregation functions. CSPA-W prefers Ω_{min} slightly more than others, while DICLENS-W undoubtedly the Ω_{CLK}. EAC-W achieves optimal results the most frequently with Ω_{CLK}, too. Results from the `REAL` domain show that all the options are more or less equally represented. Perhaps CSPA-W shows tendency to work better with Π_{PPCA} and Ω_{CLK} than others.

If we consider the frequencies of the reduction functions across data families (green bars), we see that the sum of frequencies of reduction functions other than Π_{none} is well above the latter. It may indicate that the reduction step improves the performance of consensus.

5.4.4 Cluster ensembles vs. single-clustering algorithms

Finally, let us answer the question, whether there is any improvement in performance when using the cluster ensemble approach or not. We extract the experimental results for the single-clustering algorithms with their optimal parameter's configuration for each dataset using the CVI protocol. Moreover, in their case, datasets were not

Table 5.2

ARI scores on the `Complex2D` datasets. The best column-wise scores, considering full-precision, are marked with •. The scores of EAC-W, CSPA-W, and DICLENS-W that are better or equal than EAC, CSPA, and DICLENS, respectively, are marked with ◦.

	AL	SL	CL	KM	Sp	gSOM
single	0.866	0.954	0.804	0.855	0.982	0.856
EAC	0.870	0.954	0.805	0.948	0.982	0.845
JWEAC	0.870	0.954	0.805	0.946	0.982	0.852
EAC-W	0.893 ◦	0.954 ◦	0.808 ◦	0.952 ◦	0.983 ◦	0.890 ◦
PAC	0.873	0.958	0.810	0.948	0.982	0.862
WEA	0.880	0.962	0.814	0.944	0.982	0.828
LCE	0.870	0.954	0.804	0.951	0.982	0.852
HGPA	0.633	0.314	0.542	0.989 •	0.918	0.918
MCLA	0.903	0.993	0.824	0.985	0.989	0.949
CSPA	0.928 •	0.999	0.862 •	0.981	0.991 •	0.961 •
CSPA-W	0.911	0.999 •	0.830	0.981 ◦	0.991	0.959
DICLENS	0.867	0.955	0.804	0.918	0.986	0.864
DICLENS-W	0.868 ◦	0.960 ◦	0.809 ◦	0.921 ◦	0.986	0.903 ◦

subsampled by features and the K_T scheme for the selection of cluster number is considered. For consensus functions we used the best ensemble-generation settings, i.e. the number of features to select from the dataset and the selection of K, for every dataset. We measure the average ARI score over datasets and over multiple independent runs.

In Tab. 5.2, 5.3, and 5.4 we show the results on `Complex2D`, `GENE`, and `REAL` datasets, respectively. The best scores of each clustering algorithm, considering the full-precision of numbers, are highlighted with •. We also marked with ◦ the cases where the scores of EAC-W, CSPA-W, and DICLENS-W are better or equal than EAC, CSPA, and DICLENS, respectively. So, the first row in the tables corresponds to the scenario that the clustering algorithms in the columns are regarded as the single-clustering methods. Other rows are populated by the results of consensus functions that consolidate the partitions made by the clustering algorithms in columns.

Our findings are that the great majority of cluster ensemble techniques achieve higher average ARI score than the single-clustering algorithms. We also see that it is im-

Table 5.3

ARI scores on the GENE datasets. The best column-wise scores, considering full-precision, are marked with •. The scores of EAC-W, CSPA-W, and DICLENS-W that are better or equal than EAC, CSPA, and DICLENS, respectively, are marked with ◦.

	AL	SL	CL	KM	Sp	gSOM
single	0.033	0.015	0.055	0.174	0.157	0.140
EAC	0.045	0.022	0.082	0.205	0.182	0.175
JWEAC	0.045	0.022	0.082	0.205	0.183	0.176
EAC-W	0.059 ◦	0.035 ◦	0.101 ◦	0.200	0.189 ◦	0.182 ◦
PAC	0.045	0.021	0.087	0.205	0.185	0.185
WEA	0.052	0.025	0.098	0.199	0.175	0.171
LCE	0.047	0.023	0.087	0.204	0.185	0.176
HGPA	0.113	0.021	0.216	0.234	0.215	0.211
MCLA	0.127	0.073	0.246 •	0.280 •	0.253 •	0.268 •
CSPA	0.136 •	0.108	0.200	0.225	0.223	0.206
CSPA-W	0.121	0.109 •	0.195	0.229 ◦	0.214	0.198
DICLENS	0.114	0.066	0.193	0.238	0.221	0.193
DICLENS-W	0.112	0.070 ◦	0.192	0.239 ◦	0.236 ◦	0.200 ◦

portant which clustering algorithm is employed in the process of ensemble generation – for instance, take HGPA consensus function on the Complex2D datasets (Tab. 5.2). When combined with AL, SL, and CL, it achieved considerably lower score than with KM, Sp, and gSOM.

5.5 Conclusion

We discussed a cluster ensemble framework in this chapter for it is a well-established approach towards the accurate and robust cluster analysis. Our focus was on consensus functions based on the evidence accumulation principle (EAC). We are especially interested in the weighted cluster ensembles, thus we provided with the detailed review of the weighted variants of the co-association matrix. The latter is the core of the methods based on the evidence accumulation. Among the weighting schemes, we addressed the partition relevance analysis (PRA) that utilizes the cluster validity indices (CVIs) for validating the ensemble partitions in order to weight them accordingly to

Table 5.4

ARI scores on the REAL datasets. The best column-wise scores, considering full-precision, are marked with •. The scores of EAC-W, CSPA-W, and DICLENS-W that are better or equal than EAC, CSPA, and DICLENS, respectively, are marked with ◦.

	AL	SL	CL	KM	Sp	gSOM
single	0.093	0.038	0.214	0.333	0.382	0.275
EAC	0.115	0.054	0.239	0.341	0.389	0.285
JWEAC	0.116	0.054	0.243	0.341	0.389	0.292
EAC-W	0.126 ◦	0.054 ◦	0.237	0.339	0.388	0.292 ◦
PAC	0.155	0.065	0.243	0.343	0.389	0.308
WEA	0.138	0.058	0.233	0.330	0.390	0.232
LCE	0.131	0.052	0.244	0.344	0.390	0.301
HGPA	0.336	0.192	0.366	0.394	0.415	0.382
MCLA	0.354	0.173	0.352	0.393	0.423 •	0.371
CSPA	0.406 •	0.293	0.397 •	0.397 •	0.411	0.382 •
CSPA-W	0.393	0.293 •	0.396	0.394	0.411 ◦	0.381
DICLENS	0.254	0.139	0.273	0.352	0.397	0.304
DICLENS-W	0.316 ◦	0.150 ◦	0.292 ◦	0.347	0.397	0.318 ◦

their relevance. Originally, the PRA consists of the unification and the aggregation step that result in the weights used by the consensus function.

We proposed the enhancement of the PRA introducing the reduction step that alleviates the whole weighting process by removing the redundant CVIs' assessments. More specifically, our proposal addresses the field of CVI ensembles, where we extended the studies of Vendramin et al. [225] and Jaskowiak et al. [49] with the following contributions:

- our system automatically determines the required size of CVIs subset using an intrinsic-dimensionality estimator;
- selection of the CVIs is done using the unsupervised feature selection and extraction methods;
- we systematically assessed multiple options of the unification, reduction and aggregation functions;

The PRAr step effectiveness is evaluated using the modification of three well-known consensus functions: EAC, CSPA, and DICLENS. The comprehensive experimental study was conducted including multitude of different configurations for ensemble-generation and for the PRAr step itself. The same three families of datasets were considered as in Chapters 4 and 3 using 6 single-clustering algorithms and 12 consensus functions. In this chapter we utilized the deliverables from Chapters 4 and 3 in the cluster ensemble context. The proposed gSOM algorithm was used for ensemble generation and the proposed DNs index in the PRAr step as one of 34 CVIs.

The obtained results show the boost in performance when using cluster ensemble approach. However, there are very small differences between different consensus functions, often statistically non-significant. We found it interesting, but in some way also quite expected as we experimented with the consensus functions relatively similar to each other by their design – except for the HGPA and MCLA, they all are founded on the principle of evidence accumulation. Moreover, the introduction of PRAr step proves to be beneficial as the methods with enabled PRAr step often achieve higher scores than the variants of those methods without the PRAr. Note that statistical test could not tell if methods with PRAr step are significantly better or not. In-depth analysis of the influences of the ensemble-generation techniques was carried out using the Bayesian modelling that is proposed as the alternative methodology of evaluating the performance of machine-learning systems with multiple levels of processing pipeline.

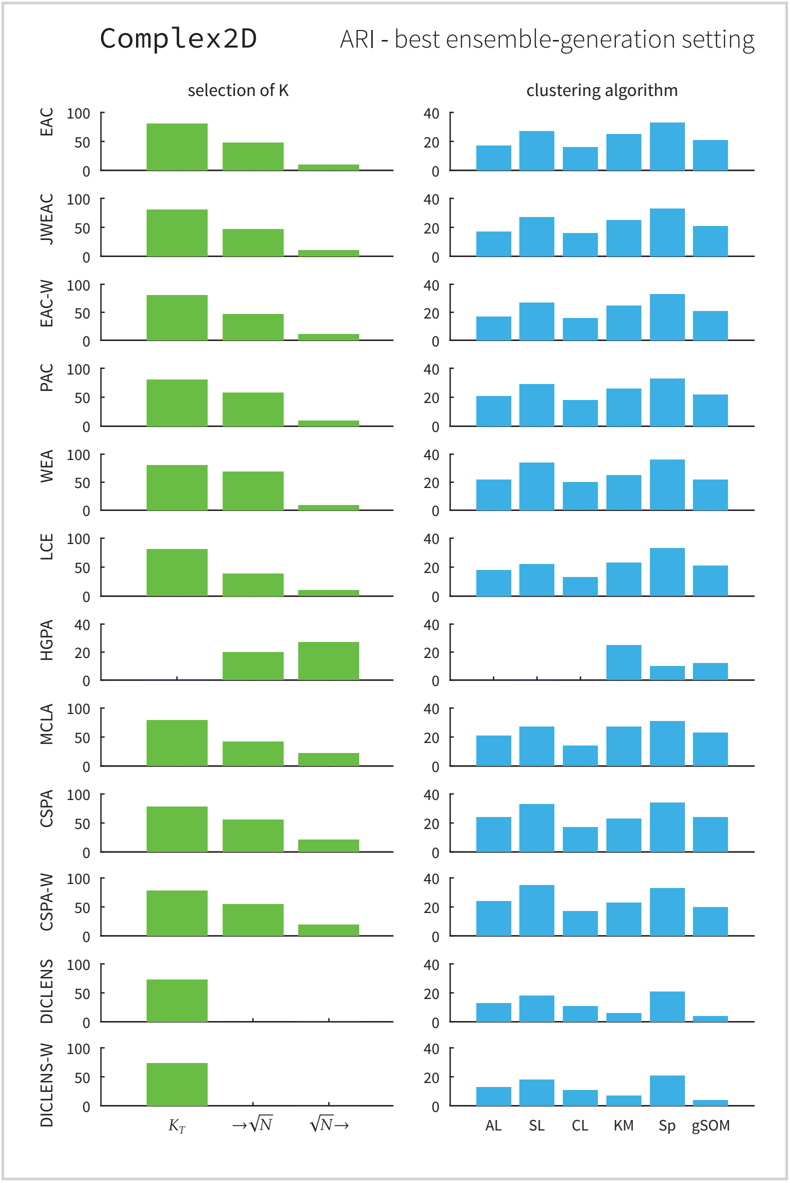


Figure 5.10

The number of times an option of ensemble generation setting participates in the best configuration measured by the average of the ARI score across multiple repetitions on the `Complex2D` datasets.

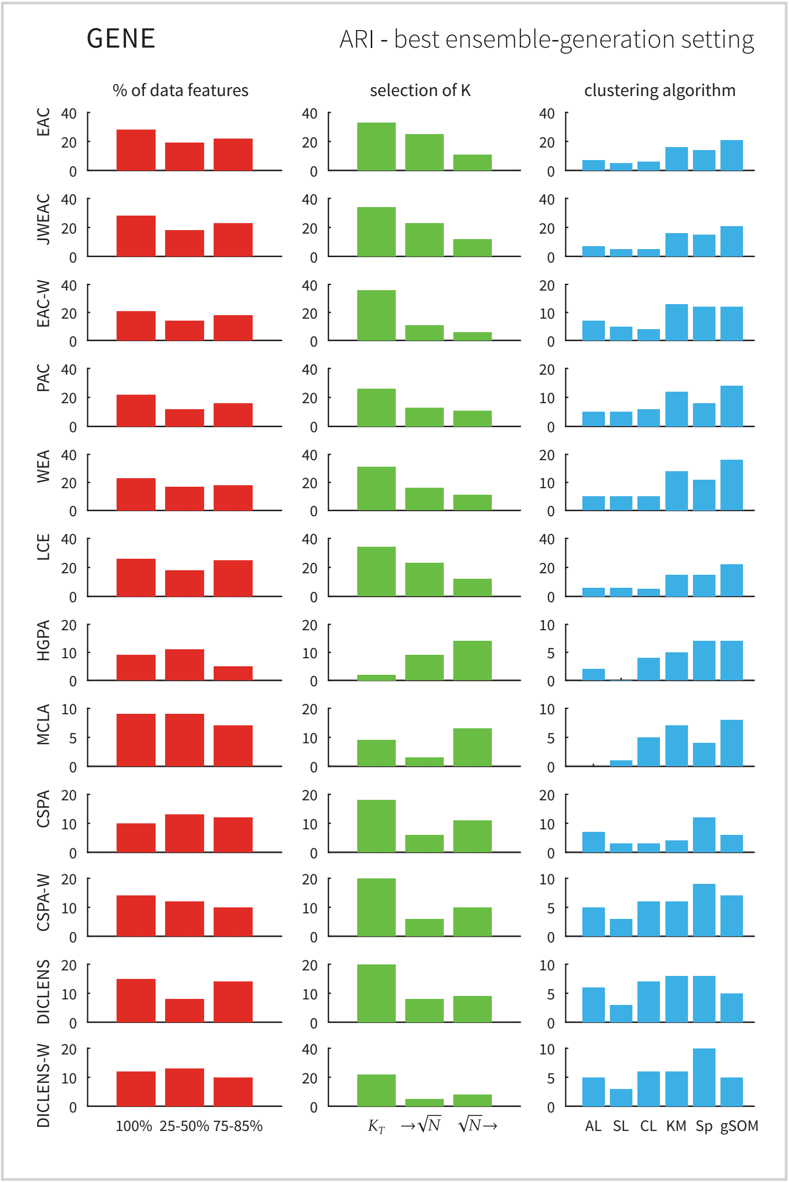


Figure 5.11

The number of times an option of ensemble generation setting participates in the best configuration measured by the average of the ARI score across multiple repetitions on the GENE datasets.

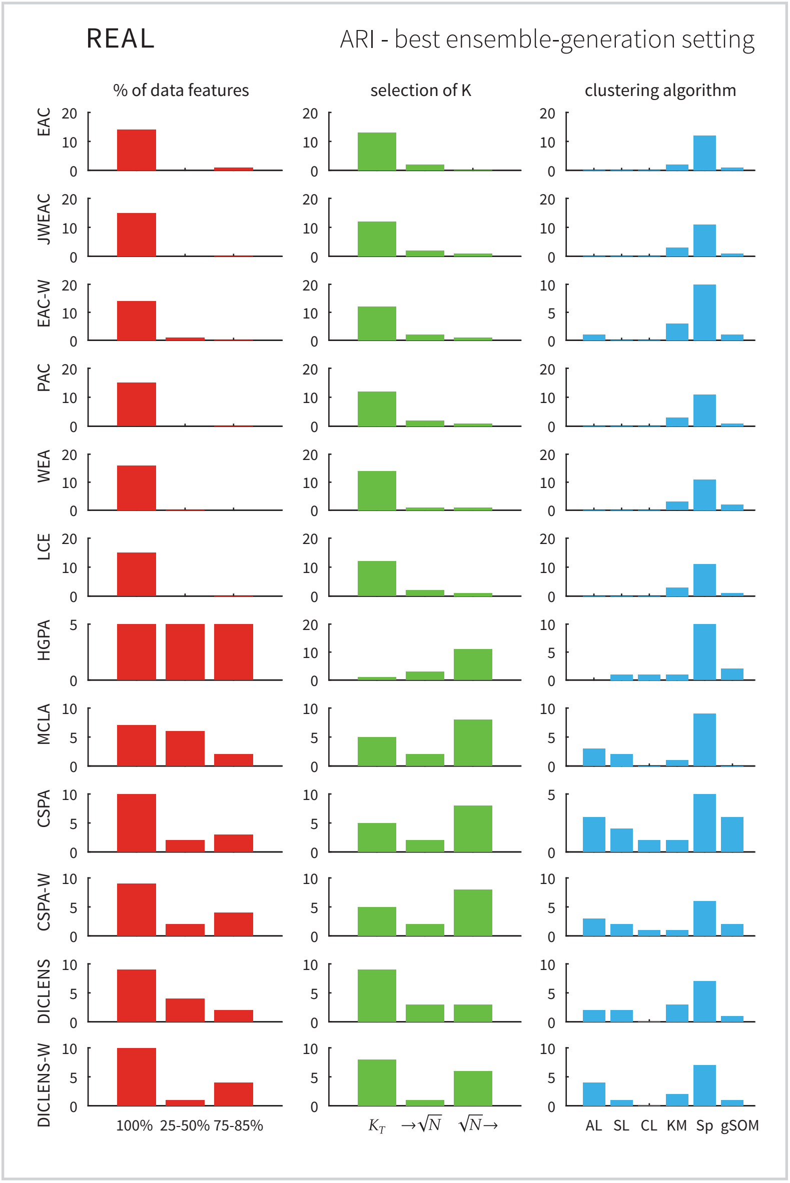


Figure 5.12

The number of times an option of ensemble generation setting participates in the best configuration measured by the average of the ARI score across multiple repetitions on the REAL datasets.

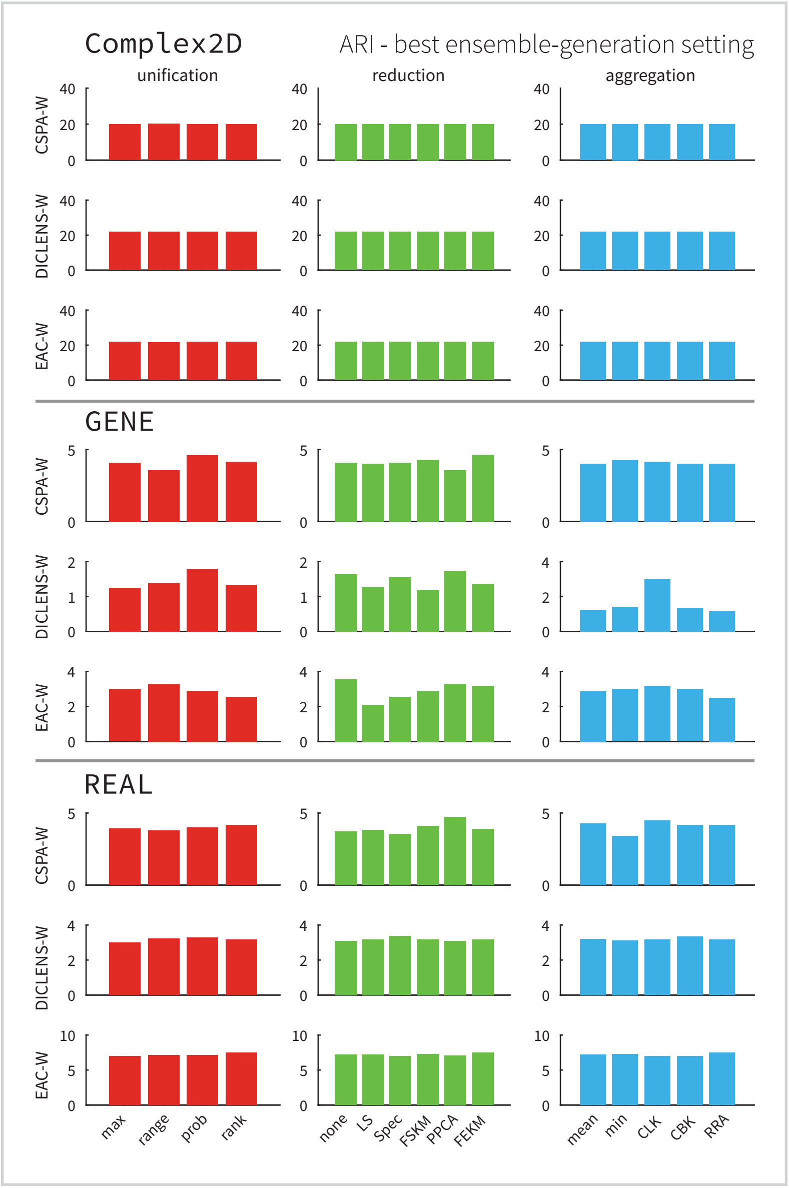


Figure 5.13

The number of times an option of ensemble-generation setting with enabled PRAr step participates in the best configuration measured by the average of the ARI score.

6

Conclusion

With this chapter we round up our story about data clustering. We have walked through the complete process of a state-of-art clustering system using cluster ensembles, including the data generation tool. We have discussed cluster analysis using single-clustering methods and cluster validation separately – then we have considered them as parts of the weighted cluster ensemble framework. We designed and performed an extensive comparison experiment in all the steps of cluster ensemble pipeline using rigorous statistical testing for differences; we used a wide range of datasets' types from synthetic to real-world data.

We addressed the field of cluster validation and motivated its usage as essential in cluster analysis. We proposed an evolved version of the Dunn's internal validity index. Our proposal is called the DNs index and is based on the shortest paths between the data points considering the Gabriel graph on the data. It features an enhancement in a form of penalization for large number of clusters, which are in practice undesirable or unexpected. A lot has been already done in this field of research with numerous validity indices devised. From the literature we selected 33 other validation indices and compared them with the DNs index on the synthetic as well as on real-world datasets of various types. To the best of our knowledge it is the first study that systematically addresses the performance of cluster validity indices (CVIs) in a case of datasets with linearly non-separable clusters. We have also discussed the issue how to measure the performance of validity index, so how to validate the validator. Based on the existing body of literature we considered four complementary methodologies, so the reader is aware of the differences among them. Our study is one of the largest in the literature with respect to the number of compared indices, datasets and performance measures used. We saw from the results of the experiment that the proposed DNs index is always better than its ancestor, the Dunn's index. The improvement in performance is statistically significant for the majority of the cases. Moreover, the DNs index is consistently placed among the best-performing indices.

Furthermore, we reviewed and pointed out the most influential works in the context of two-level cluster analysis based on the self-organizing map (SOM). We tightly integrated the gravitational clustering approach with the SOM and developed the gSOM algorithm that is able to automatically estimate the number of clusters in the data. It has low time complexity and is able to produce a hierarchy of clusters, helpful for a data-miner to better grasp the structure of the data being processed. The gSOM algorithm proved to be competent when comparing its performance against other SOM-based and conventional clustering methods – the most significantly on genetic data.

The cluster ensemble framework is a well-established approach towards the accurate and stable cluster analysis. In this thesis we are especially interested in the weighted cluster ensembles, thus we provided the detailed review of the weighted variants of the evidence accumulation principle that is very common among the consensus functions. We built our research on the promising results of the studies that proposed the weighting of the partitions using the cluster validity indices. We proposed the enhancement of such approach called the partition relevance analysis with reduction step (PRAr). We showed empirically that reducing the large initial set of validity indices alleviates the whole weighting process by removing the redundant CVIs' assessments before the aggregation step. We developed the reduction procedure that automatically determines the required size of CVIs subset using an intrinsic-dimensionality estimator and then applies the unsupervised feature selection or extraction methods to actually reduce the matrix of unified CVIs scores. We plugged the PRAr into the three consensus functions, i.e. EAC, CSPA, and DICLENS, and conducted a comprehensive experimental study including different configurations of ensemble-generation and of the PRAr step itself. We can summarize our findings as follows:

- The cluster ensemble approach boosts the average performance of single-clustering algorithms.
- Differences between twelve consensus functions in comparison are found to be quite non-discriminative, in the most cases statistically non-significant. It was expected, as the majority of them is founded on the evidence accumulation.
- The introduction of the reduction step in the partition relevance analysis (PRAr) proves to be beneficial – the methods EAC-W, CSPA-W, and DICLENS-W with enabled PRAr step often showed equal or better performance than their variants without the PRAr although not significantly so.

6.1 Future Research Directions

What we have learned during all these years of developing and improving cluster analysis algorithms is that this is a never-ending story. Data clustering is an ill-posed problem meaning that there are many possible solutions and interpretations, depending on how you define *good* clustering [4]. Thus, there is always some work to do – let us motivate a few interesting avenues for research to go next.

One of disadvantages of the presented gSOM algorithm is its sensitivity and dependency on the parameters. We found the recently proposed heuristics for the parameterless gravitational clustering algorithm by Gomez et al. [165] very inspiring and we are looking forward to integrating them into the context of gSOM. We are satisfied with the results of gSOM on genetic data – however, we considered only the Euclidean distance between data samples. We believe it would be beneficial to implement also other measures of (dis)similarity like the correlation coefficient and cosine distance as suggested by the recent study of selecting the appropriate distances for gene expression data clustering by Jaskowiak et al. [244].

Yet another idea for the future work is connected to genetic data. When we evaluated 34 selected cluster validity indices on the GENE datasets, we noticed poor performance scores among all the competitors. The reason probably lies in the fact that these CVIs are *general purpose* indices, not particularly tailored to tackle the problems of genetic data. Datta and Datta proposed few such indices for validation of gene expression data. They utilized external information about the functional classes of genes [48, 245]. We are curious how these CVIs compare to general-purpose ones we addressed and whether there is a possibility to improve them with the information about the functional class of a certain gene.

What about the ensemble-clustering perspectives? One possible speculation based on our experimental results is that the evidence accumulation approach is somewhere near its limits of performance. We have assessed a dozen of representative algorithms and found almost no significant differences among them. However, the weighted ensembles with PRAr proved they are valuable in achieving better results. We think that additional effort should be made to propagate this approach to cluster ensembles based on the median partition search [10, 197].

Supplement to chapter "Gravitational Clustering of Self-organizing map"

A

Table A.1

The number of clusters in the partitions for the best parameters' configurations of the algorithms with automatic number of clusters detection on the `Complex2D` datasets. To save space, we omitted the `Complex2D` prefix in the datasets names. The best parameters' configurations are determined by three protocols using the external (eCVI) and internal (CVI) indices, and cross-validation (CV). In the last row we show the number of matching with true number of clusters. The highest matching rates are underlined for each protocol. The ARI index is used for the eCVI and CV protocols, and the DNs index for the CVI protocol.

`Complex2D`	Clusot*			SOMSpec*			SOMStar*			SOMKm*			gSOM*		
dataset	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI
`K2_D50_L0`	2	2	2	2	2	2	2	7	2	2	2	2	2	2	2
`K3_D50_L0`	4	4	4	3	2	3	3	5	4	3	3	3	3	3	3
`K6_D50_L0`	5	4	2	6	6	6	6	7	7	6	6	6	6	6	6
`K8_D50_L0`	6	3	2	8	9	8	8	9	4	8	8	8	8	8	8
`K2_D30_L0`	2	2	2	2	2	2	4	7	6	15	10	15	2	2	2
`K3_D30_L0`	4	2	2	3	3	3	4	5	5	3	3	3	3	3	3
`K6_D30_L0`	4	2	2	6	6	6	6	6	8	6	6	6	6	6	6
`K8_D30_L0`	2	2	2	8	9	8	10	10	10	8	8	9	8	8	8
`K2_D15_L1`	2	3	2	2	20	7	6	6	6	16	16	16	2	7	2
`K3_D15_L1`	3	3	2	3	3	2	5	6	6	4	9	4	3	5	2
`K6_D15_L1`	2	4	2	6	17	6	7	7	4	23	23	23	6	6	6
`K8_D15_L1`	6	2	2	8	9	8	10	8	8	23	22	22	11	8	2
`K2_D10_L1`	3	2	2	2	11	9	4	6	9	9	9	10	2	7	2
`K3_D10_L1`	2	2	2	3	9	3	6	6	8	2	2	2	3	3	2
`K6_D10_L1`	3	3	2	6	5	7	5	5	5	23	21	23	6	5	3
`K8_D10_L1`	2	3	2	12	9	2	9	7	7	23	4	4	10	4	4
`K3_D12_L2`	2	2	2	3	10	3	6	8	6	23	19	19	3	3	2
`K6_D12_L2`	3	5	2	6	12	2	8	9	9	7	23	7	9	5	5
`K8_D12_L2`	4	3	2	8	7	7	13	8	7	14	20	14	10	3	2
`K3_D10_L2`	2	3	2	6	6	6	3	10	10	3	3	6	3	3	3
`K6_D10_L2`	2	2	2	6	11	2	7	6	7	3	3	3	7	2	4
`K8_D10_L2`	2	2	2	8	2	2	11	7	4	9	22	9	12	4	3
matches with K_T	4	5	4	20	6	12	6	4	2	8	8	6	16	13	12

Table A.2

The number of clusters in the partitions for the best parameters' configurations of the algorithms with automatic number of clusters detection on the `GENE` datasets. To save space, we omitted the `Complex2D` prefix in the datasets names. The best parameters' configurations are determined by three protocols using the external (eCVI) and internal (CVI) indices, and cross-validation (CV). In the last row we show the number of matching with true number of clusters. The highest matching rates are underlined for each protocol. The ARI index is used for the eCVI and CV protocols, and the CH index for the CVI protocol.

`GENE`	Clusot*			SOMSpec*			SOMStar*			SOMKm*			gSOM*		
dataset	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI
`bladderC.`	2	3	2	6	4	2	3	2	3	8	8	6	3	11	2
`breastC.`	2	2	2	3	4	2	4	2	2	7	4	3	10	3	2
`breastC.T.`	3	2	2	3	3	2	2	2	2	11	3	2	6	3	2
`carcinomas`	5	2	2	12	2	2	9	3	2	15	15	15	16	2	2
`centNrvSys.`	3	3	3	5	4	2	2	2	3	7	7	3	8	8	2
`endometr.C.`	2	2	2	3	8	2	3	3	3	9	9	9	6	9	2
`glioblast.`	4	3	2	5	2	2	2	3	2	8	7	7	7	5	2
`gliomagen.`	3	3	2	2	4	2	3	4	2	8	8	6	2	15	2
`gliomas`	4	2	3	2	6	2	3	2	3	9	9	9	4	5	2
`hepatocell.C.`	2	2	2	3	2	2	3	3	2	14	14	2	6	2	2
`leukemia-1`	6	2	2	10	11	2	2	2	2	4	16	3	5	12	2
`leukemia-2`	2	2	2	3	7	2	4	2	2	9	9	9	2	16	2
`leukemia-3`	2	2	2	3	6	2	2	3	3	9	9	9	5	16	2
`lungT.-1`	3	3	2	6	8	2	3	4	4	15	15	11	7	11	2
`lungT.-2`	2	2	2	2	7	2	3	2	2	8	9	9	6	9	2
`lymphoma-1`	4	2	2	3	5	2	2	2	3	7	6	6	12	14	2
`lymphoma-2`	2	3	2	3	7	2	4	2	2	9	8	9	2	13	3
`melanoma`	5	2	2	4	4	2	2	2	2	7	7	7	20	9	2
`mesothel.`	4	2	2	2	14	2	4	4	4	14	14	14	2	20	2
`multitissue`	2	2	3	9	10	2	5	4	5	19	19	19	9	8	2
`prostateC.-1`	2	2	2	2	6	2	3	2	2	10	8	10	4	16	2
`prostateC.-2`	3	2	2	4	5	2	8	2	2	2	2	2	11	6	2
`prostateC.-3`	2	2	2	3	9	2	3	2	2	11	11	11	9	8	2
`roundBluecellT.`	2	4	2	10	4	2	9	2	3	10	10	10	14	12	2
`serratedC.`	3	4	2	6	7	2	2	2	2	7	7	7	2	12	2
matches with K_T	$\underline{8}$	$\underline{13}$	11	2	2	$\underline{12}$	$\underline{8}$	12	11	0	0	2	7	1	11

Table A.3

The number of clusters in the partitions for the best parameters' configurations of the algorithms with automatic number of clusters detection on the `REAL` datasets. To save space, we omitted the `Complex2D` prefix in the datasets names. The best parameters' configurations are determined by three protocols using the external (eCVI) and internal (CVI) indices, and cross-validation (CV). In the last row we show the number of matching with true number of clusters. The highest matching rates are underlined for each protocol. The ARI index is used for the eCVI and CV protocols, and the CH index for the CVI protocol.

`REAL`	Clusot*			SOMSpec*			SOMStar*			SOMKm*			gSOM*		
dataset	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI	eCVI	CV	CVI
`breastC.`	3	3	2	2	2	2	3	3	3	2	21	2	2	2	2
`contrac.`	2	4	2	2	10	2	5	3	3	8	10	8	2	6	2
`fertility`	7	2	2	4	9	2	5	2	3	10	10	10	2	5	2
`glass`	3	3	3	5	7	6	5	2	2	15	8	8	7	5	2
`ionosph.`	2	3	2	5	10	2	5	4	4	19	19	19	6	5	2
`iris`	2	2	2	3	2	2	2	3	2	2	13	2	4	4	2
`laryngeal3`	4	2	2	4	13	4	3	5	10	19	19	19	6	3	2
`letterABC`	3	3	3	5	10	5	6	12	4	25	25	25	8	7	3
`respirat.`	2	2	2	2	2	2	2	3	2	9	10	10	2	2	2
`segment.`	3	2	2	12	6	2	11	5	2	14	15	15	20	5	2
`voice_3`	2	3	2	3	3	2	10	7	2	16	16	16	3	3	3
`weaning`	4	4	2	10	14	2	7	7	2	18	18	18	9	2	2
`wine`	4	2	2	3	7	3	3	3	3	14	14	14	4	3	2
`wineRed`	2	2	2	6	20	10	3	7	2	24	25	25	7	4	2
`yeast`	2	2	2	6	6	8	9	3	3	8	8	8	11	5	7
matches with K_T	4	4	4	$\underline{6}$	3	$\underline{6}$	4	3	2	1	0	1	5	$\underline{6}$	$\underline{6}$

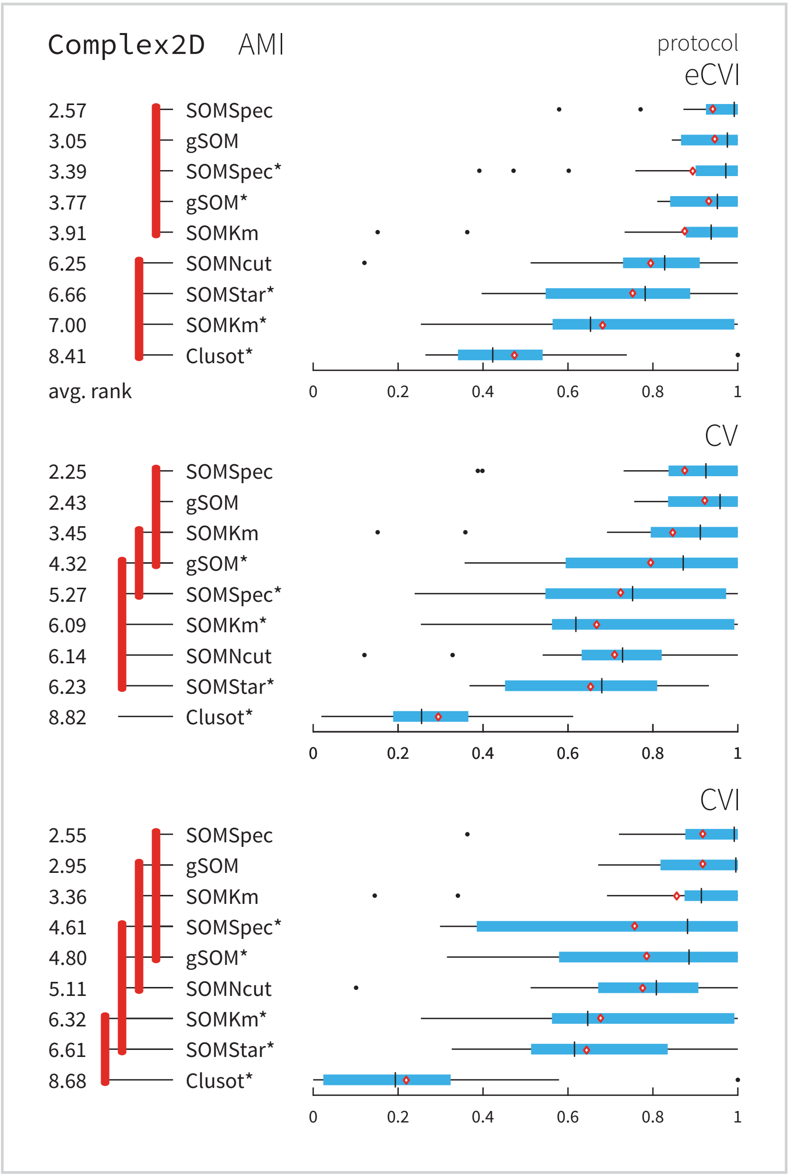


Figure A.1

Average ranks of algorithms based on the AMI score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

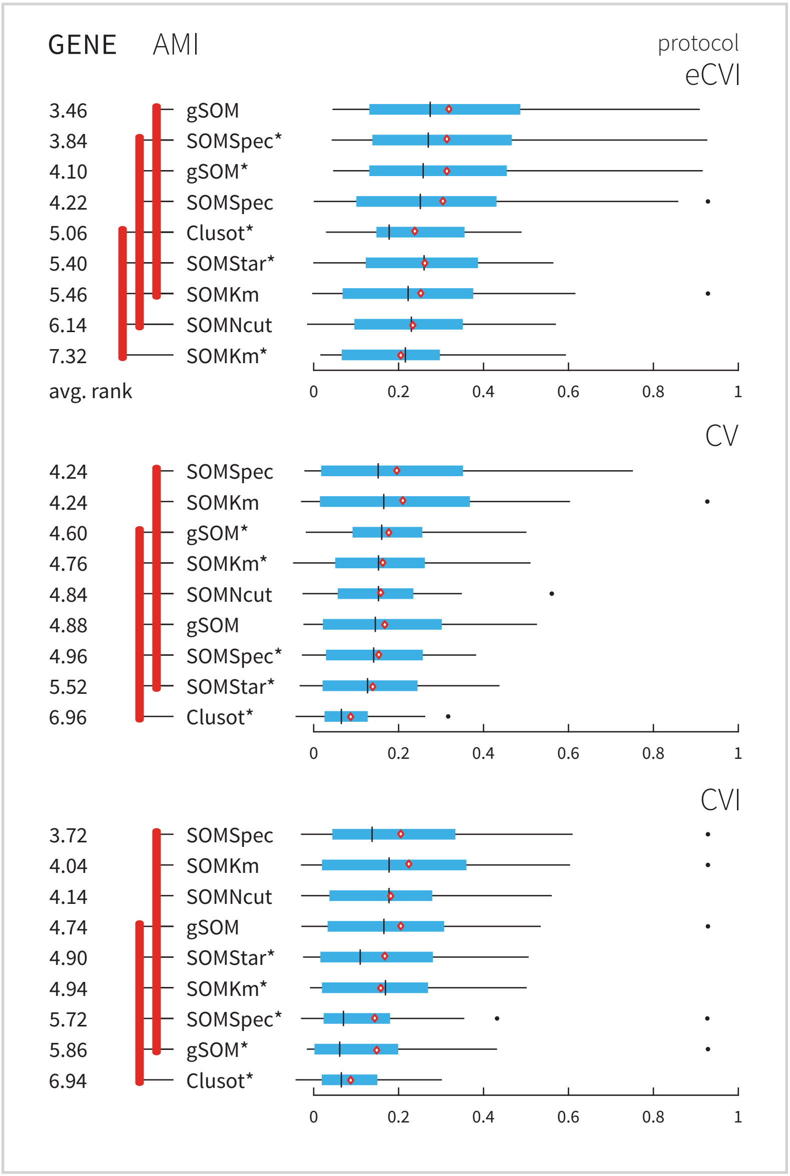


Figure A.2

Average ranks of algorithms based on the AMI score across all the GENE datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

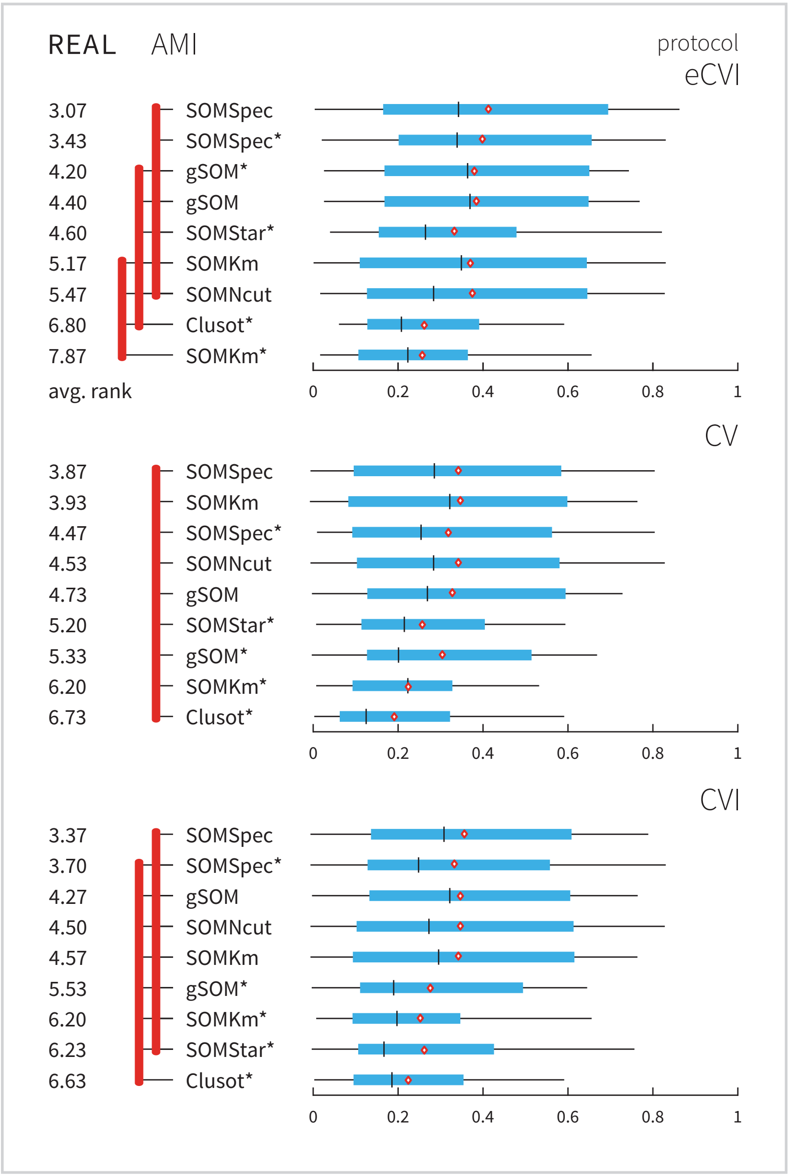


Figure A.3

Average ranks of algorithms based on the AMI score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$, except for the CV protocol, where Friedman's test does not reject the null hypothesis of equal ranks. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

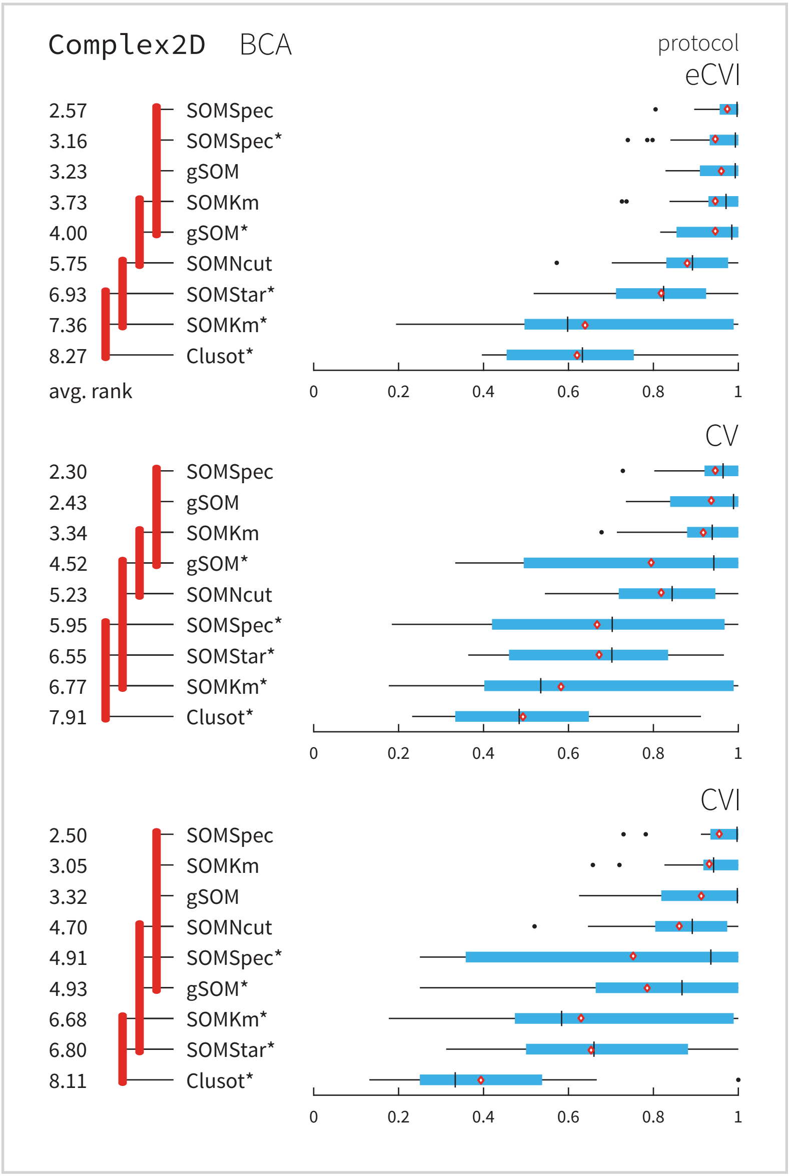


Figure A.4

Average ranks of algorithms based on the BCA score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

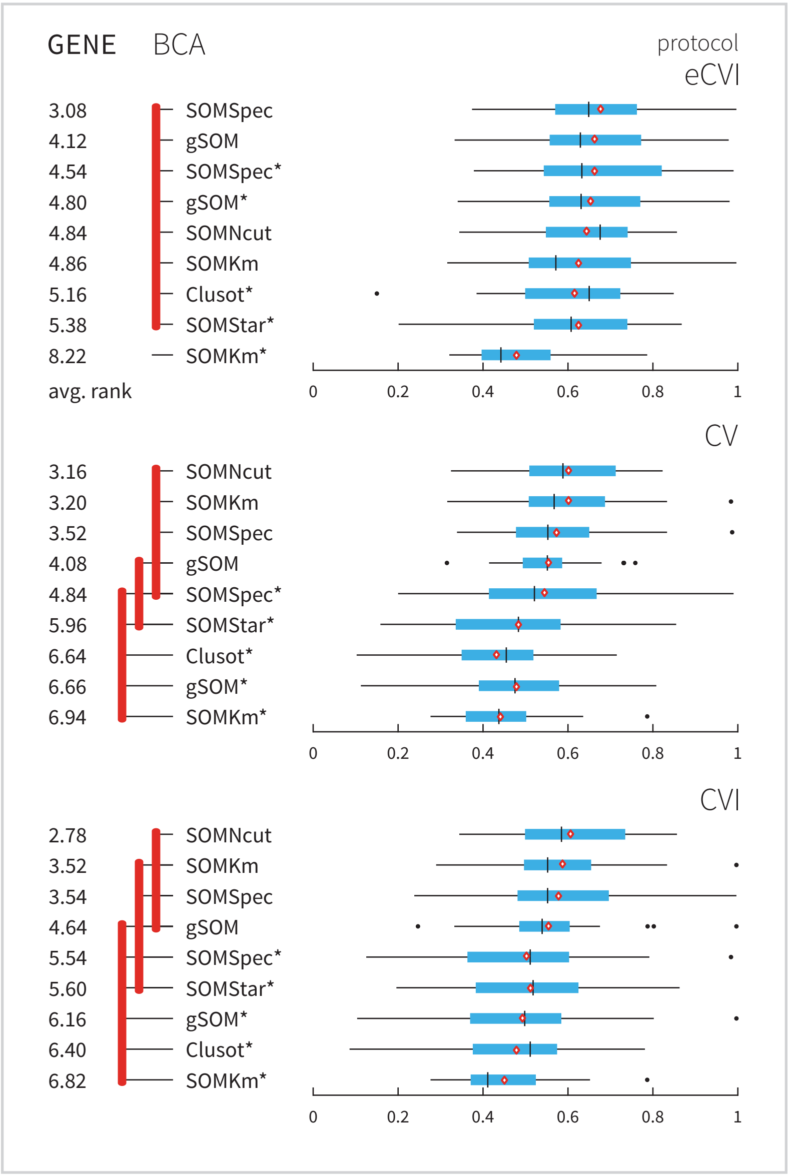


Figure A.5

Average ranks of algorithms based on the BCA score across all the GENE datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

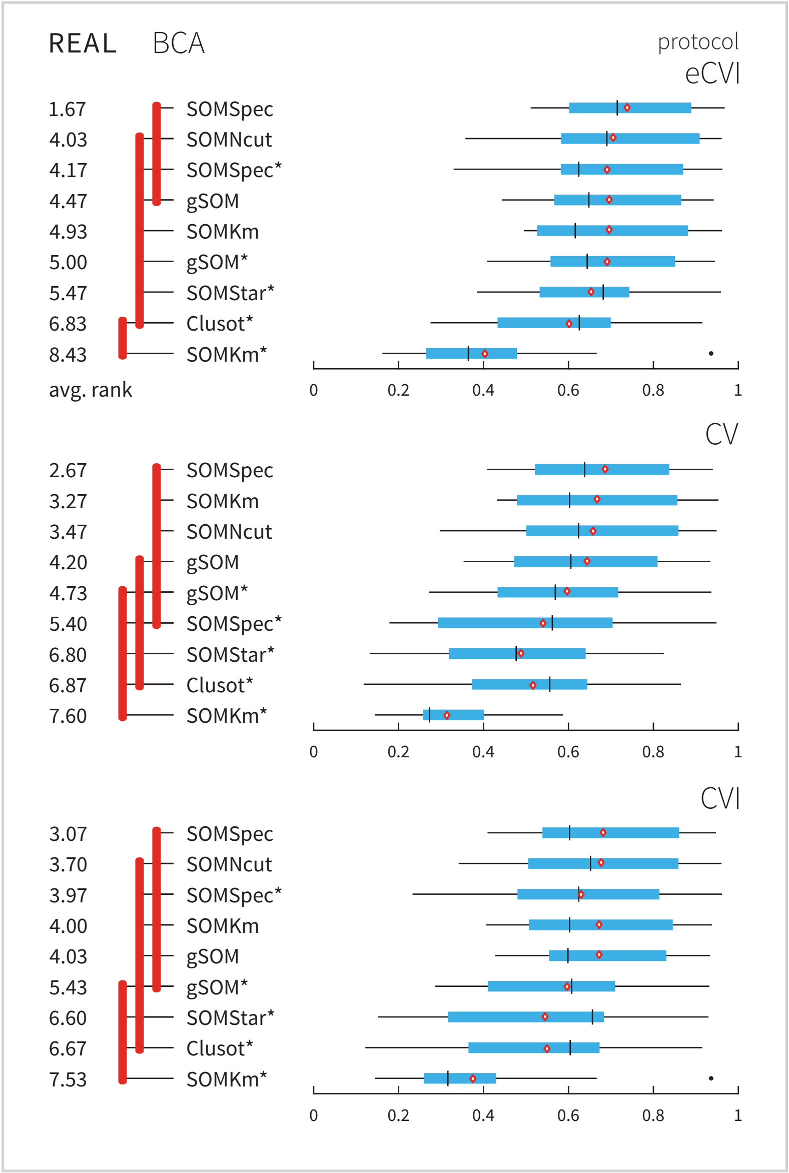


Figure A.6

Average ranks of algorithms based on the BCA score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

Table A.4

The best parameter's configurations on the `Complex2D` datasets considering the ARI score and the eCVI protocol. In the last row there are configurations that are best on average over all the datasets. The symbol □ is used for rectangular SOM grid (*g*) and ⬢ for hexagonal grid.

`Complex2D`	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	`res`	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	`kNN`	S	g	`kNN`	S	σ
`K2_D50_L0`	1	⬢	0	0.1	0.5	2	□	5.0E-4	0	2	□	5.0E-4	0.01	2	□	2	⬢	0.5	□	2	□	1	2	□	2	0.5	2
`K3_D50_L0`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	⬢	0.5	□	2	□	1	2	□	2	1	3
`K6_D50_L0`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	□	0.5	□	2	□	1	2	□	2	0.5	0.5
`K8_D50_L0`	2	□	0	0.5	0.5	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	⬢	1	□	2	□	1	2	□	2	2	2
`K2_D30_L0`	0.5	□	0	0.1	0.5	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	□	0.5	□	2	□	1	2	□	2	0.5	2
`K3_D30_L0`	0.5	⬢	0	0.3	0.5	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	⬢	1	□	2	□	1	2	□	2	0.5	1.5
`K6_D30_L0`	0.5	□	0.2	0.5	0.25	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	□	2	⬢	0.5	□	2	□	1	2	□	3	1	0.5
`K8_D30_L0`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	1.0E-4	0.01	2	⬢	2	⬢	2	□	2	□	1	2	□	2	2	1.5
`K2_D15_L1`	0.5	⬢	0	0.1	0.25	0.5	□	5.0E-4	0.001	0.5	□	5.0E-4	0.005	2	⬢	2	⬢	1	□	2	□	1	1	□	7	0.5	0.5
`K3_D15_L1`	0.5	□	0	0.1	0.5	2	□	5.0E-4	0.01	2	□	1.0E-3	0.05	2	□	2	□	0.5	⬢	1	□	2	2	□	9	0.5	1
`K6_D15_L1`	2	⬢	0	0.1	0.5	2	□	1.0E-4	0.005	2	□	1.0E-4	0.01	0.5	□	1	⬢	1	⬢	2	□	2	2	□	2	1	1
`K8_D15_L1`	0.5	□	0.2	0.5	0.25	2	⬢	1.0E-4	0.05	2	⬢	1.0E-4	0.05	2	□	0.5	⬢	2	⬢	2	□	2	2	⬢	11	1	0.5
`K2_D10_L1`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	5.0E-4	0.01	0.5	⬢	2	⬢	0.5	⬢	0.5	⬢	15	1	□	9	1	2
`K3_D10_L1`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	1.0E-4	0.005	0.5	⬢	2	⬢	1	⬢	1	□	2	1	□	3	2	2
`K6_D10_L1`	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-5	0.01	2	⬢	5.0E-5	0.01	2	□	0.5	⬢	0.5	□	2	⬢	9	2	⬢	15	0.5	0.5
`K8_D10_L1`	0.5	□	0	0.1	0.5	2	□	5.0E-5	0	2	□	5.0E-5	0.05	1	⬢	0.5	□	0.5	□	1	□	2	2	□	1	2	1
`K3_D12_L2`	1	⬢	0.2	0.3	0.1	2	□	5.0E-5	0.01	2	□	5.0E-5	0.01	0.5	□	0.5	⬢	0.5	□	0.5	⬢	15	0.5	⬢	5	0.5	0.5
`K6_D12_L2`	1	□	0.2	0.3	0.05	2	□	1.0E-3	0.01	2	□	5.0E-5	0.05	2	□	1	⬢	0.5	⬢	2	□	5	2	□	5	2	1.5
`K8_D12_L2`	1	⬢	0.2	0.5	0.5	2	□	5.0E-5	0.05	2	⬢	1.0E-5	0.001	2	□	1	⬢	0.5	⬢	2	⬢	15	2	□	15	2	0.5
`K3_D10_L2`	2	⬢	0	0.1	0.5	0.5	⬢	5.0E-5	0.001	0.5	⬢	5.0E-5	0.001	0.5	⬢	2	⬢	2	⬢	2	□	1	2	□	3	0.5	3
`K6_D10_L2`	0.5	□	0	0.1	0.5	2	□	5.0E-5	0.05	2	□	1.0E-4	0.1	2	⬢	0.5	⬢	1	⬢	2	⬢	3	1	□	11	1	0.5
`K8_D10_L2`	0.5	⬢	0	0.3	0.5	2	⬢	1.0E-4	0.05	2	□	5.0E-5	0.05	2	□	2	□	0.5	□	2	□	9	2	⬢	2	2	1
AVG	0.5	⬢	0.2	0.3	0.25	2	□	5.0E-4	0	2	□	5.0E-4	0.1	2	⬢	2	⬢	0.5	□	2	□	1	1	□	3	1	1

Table A.5

The best parameter's configurations on the GENE datasets considering the ARI score and the eCVI protocol. In the last row there are configurations that are best on average over all the datasets. The symbol □ is used for rectangular SOM grid (*g*) and ⬢ for hexagonal grid.

GENE	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	res	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	kNN	S	g	kNN	S	σ
bladderC.	0.5	⬢	0	0.1	0.25	1	□	1.0E-3	0.1	0.5	⬢	1.0E-3	0.1	0.5	⬢	0.5	⬢	1	⬢	0.5	⬢	5	2	□	5	1	0
breastC.	0.5	□	0	0.3	0.5	0.5	⬢	1.0E-4	0.1	0.5	⬢	1.0E-4	0.1	1	□	0.5	□	2	□	1	⬢	3	0.5	□	5	2	1
breastC.T.	1	⬢	0	0.1	0.1	0.5	⬢	1.0E-5	0	0.5	⬢	1.0E-5	0.005	0.5	⬢	0.5	⬢	1	⬢	2	⬢	1	2	⬢	1	1	1
carcinomas	2	□	0	0.3	0.25	1	□	5.0E-5	0.01	1	□	5.0E-5	0.01	1	□	1	□	0.5	⬢	2	□	9	2	□	5	2	0.5
centNrvSys.	2	□	0	0.9	0.25	0.5	⬢	3.0E+0	0.005	2	□	1.0E+0	1	1	□	2	⬢	2	⬢	2	⬢	1	2	□	11	2	2.5
endometr.C.	0.5	□	0	0.1	0.1	1	⬢	5.0E-5	0.01	1	⬢	5.0E-5	0.005	2	□	2	□	0.5	□	2	□	5	2	□	3	1	0
glioblast.	2	⬢	0.4	0.5	0.1	0.5	□	1.0E-5	0	0.5	□	1.0E-5	0.005	0.5	⬢	0.5	□	1	□	1	⬢	1	0.5	□	9	1	2.5
gliomagen.	2	□	0	0.1	0.1	0.5	□	5.0E-4	0	0.5	⬢	5.0E-4	0.005	1	□	2	□	2	□	1	□	1	0.5	□	3	1	2
gliomas	0.5	□	0	0.3	0.25	0.5	⬢	1.0E-4	0.005	0.5	⬢	1.0E-4	0.005	0.5	⬢	0.5	⬢	0.5	⬢	0.5	⬢	7	0.5	⬢	3	1	1
hepatocell.C.	0.5	□	0	0.3	0.25	0.5	□	5.0E-5	0.01	0.5	□	5.0E-5	0.01	0.5	⬢	0.5	□	1	□	1	⬢	2	0.5	□	5	0.5	0
leukemia-1	2	□	0.4	0.5	0.25	1	□	1.0E-4	0	0.5	⬢	1.0E-4	0.01	0.5	⬢	2	⬢	0.5	⬢	0.5	⬢	11	2	⬢	3	0.5	0
leukemia-2	1	⬢	0	0.1	0.1	0.5	⬢	1.0E-3	0.001	0.5	⬢	1.0E-3	0.005	2	□	0.5	□	1	□	2	⬢	1	1	□	5	2	1.5
leukemia-3	2	⬢	0	0.1	0.5	1	□	1.0E-4	0.005	1	□	1.0E-4	0.005	2	⬢	1	⬢	1	⬢	2	⬢	1	0.5	⬢	1	2	2.5
lungT.-1	0.5	□	0	0.3	0.25	2	□	5.0E-5	0.01	2	□	5.0E-4	0.1	2	□	0.5	□	0.5	⬢	1	⬢	1	0.5	⬢	1	1	2
lungT.-2	0.5	⬢	0	0.7	0.5	1	⬢	5.0E-5	0	1	⬢	1.0E-4	0.005	2	□	1	□	1	□	1	□	1	1	⬢	3	1	0
lymphoma-1	2	□	0	0.5	0.5	2	□	1.0E-3	0.1	2	□	1.0E-3	0.1	0.5	⬢	2	□	0.5	⬢	0.5	⬢	11	0.5	⬢	15	0.5	0
lymphoma-2	1	□	0	0.1	0.1	0.5	⬢	1.0E-3	0.001	0.5	⬢	5.0E-4	0.005	1	□	2	□	1	□	1	□	1	2	⬢	15	2	1
melanoma	2	□	0.4	0.5	0.01	2	⬢	1.0E+0	0.1	2	⬢	5.0E-5	0.1	2	□	1	□	1	□	0.5	□	15	2	⬢	3	1	2
mesothel.	2	⬢	0.4	0.5	0.1	1	□	5.0E-4	0.005	1	□	5.0E-4	0.01	1	⬢	2	□	0.5	□	1	⬢	1	1	□	1	1	1
multitissue	1	⬢	0	0.1	0.1	0.5	⬢	2.0E+0	0	0.5	⬢	3.0E+0	1	0.5	□	0.5	□	0.5	□	2	⬢	1	2	⬢	3	2	0
prostateC.-1	0.5	⬢	0	0.1	0.25	1	⬢	5.0E-5	0	1	⬢	1.0E-4	0.01	0.5	⬢	2	⬢	1	□	0.5	⬢	5	0.5	⬢	1	2	2.5
prostateC.-2	2	⬢	0	0.3	0.1	2	⬢	1.0E-3	0.1	2	⬢	5.0E-4	0.05	2	□	0.5	⬢	0.5	□	2	⬢	1	0.5	□	3	2	0
prostateC.-3	0.5	□	0	0.1	0.1	2	□	5.0E-5	0	2	□	5.0E-5	0.001	0.5	⬢	2	□	2	⬢	1	⬢	5	0.5	⬢	3	2	2
roundBluecellT.	1	⬢	0	0.1	0.1	1	⬢	1.0E-4	0.1	1	⬢	5.0E-5	0.05	0.5	⬢	1	⬢	0.5	⬢	2	□	3	2	⬢	1	2	0
serratedC.	0.5	⬢	0	0.7	0.1	0.5	⬢	5.0E-4	0	0.5	⬢	5.0E-4	0.005	1	□	1	□	0.5	⬢	1	□	1	0.5	⬢	3	0.5	1.5
AVG	2	⬢	0	0.3	0.25	1	□	1.0E-4	0.001	0.5	⬢	1.0E-3	0.5	1	□	0.5	⬢	0.5	⬢	2	⬢	1	1	□	1	1	0

Table A.6

The best parameter's configurations on the REAL datasets considering the ARI score and the eCVI protocol. In the last row there are configurations that are best on average over all the datasets. The symbol □ is used for rectangular SOM grid (g) and ⬢ for hexagonal grid.

REAL	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	res	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	kNN	S	g	kNN	S	σ
breastC.	0.5	□	0	0.1	0.5	0.5	□	1.0E-3	0	0.5	□	1.0E-3	0.05	2	□	2	□	2	⬢	1	⬢	1	1	⬢	1	0.5	1.5
contrac.	0.5	⬢	0.2	0.3	0.25	2	⬢	1.0E-2	0.05	2	⬢	5.0E-4	0.01	2	□	1	□	1	□	2	⬢	2	0.5	⬢	2	1	0.5
fertility	2	□	0.4	0.7	0.1	1	□	1.0E-1	0.5	1	□	1.0E-1	0.001	2	⬢	0.5	⬢	2	□	1	□	3	1	□	1	2	1.5
glass	0.5	□	0.4	0.5	0.25	2	⬢	1.0E-4	0.01	2	⬢	1.0E-4	0.01	0.5	□	0.5	□	0.5	□	2	□	5	1	□	3	2	1
ionosph.	0.5	⬢	0	0.1	0.5	0.5	⬢	5.0E-5	0.001	0.5	⬢	5.0E-5	0.001	2	□	2	□	0.5	□	2	□	5	0.5	⬢	1	1	0
iris	1	□	0.2	0.3	0.5	1	⬢	5.0E-5	0.01	0.5	⬢	1.0E-4	0.01	0.5	□	1	⬢	2	□	0.5	□	3	1	⬢	1	0.5	3
laryngeal3	2	□	0.2	0.9	0.25	2	□	1.0E-4	0.05	2	□	1.0E-2	0.5	2	□	1	⬢	1	⬢	1	⬢	1	0.5	⬢	1	2	1.5
letterABC	0.5	□	0.2	0.3	0.5	2	⬢	5.0E-4	0.05	2	□	1.0E-4	0.005	0.5	⬢	0.5	⬢	0.5	⬢	1	⬢	9	2	□	1	1	2.5
respirat.	0.5	□	0.2	0.3	0.25	1	⬢	1.0E-4	0	1	⬢	5.0E-4	0.001	1	⬢	0.5	□	1	⬢	2	□	3	1	⬢	11	0.5	1
segment.	0.5	□	0.2	0.7	0.25	2	⬢	1.0E-5	0	2	⬢	5.0E-5	0.05	2	⬢	1	□	2	□	2	□	3	2	□	5	2	0.5
voice_3	2	⬢	0	0.1	0.25	1	⬢	1.0E-4	0	1	⬢	1.0E-4	0.001	1	□	2	⬢	2	□	2	⬢	1	0.5	□	1	2	0.5
weaning	0.5	□	0.2	0.3	0.5	2	⬢	5.0E-5	0	2	⬢	1.0E-4	0.01	1	⬢	0.5	⬢	2	⬢	0.5	⬢	1	1	□	5	1	0
wine	1	⬢	0.2	0.5	0.25	2	□	1.0E-3	0.05	2	□	1.0E-3	0.05	1	⬢	0.5	□	1	⬢	0.5	⬢	3	1	⬢	11	1	1
wineRed	0.5	⬢	0.2	0.5	0.5	1	⬢	5.0E-5	0.01	2	⬢	5.0E-5	0.005	0.5	□	0.5	□	0.5	⬢	1	⬢	9	1	⬢	3	1	3
yeast	0.5	□	0.2	0.7	0.25	0.5	□	5.0E-5	0	0.5	□	1.0E-4	0.05	2	□	2	□	0.5	□	2	⬢	3	2	□	15	0.5	0.5
AVG	0.5	□	0	0.3	0.5	0.5	□	1.0E-3	0.01	0.5	⬢	1.0E-4	0.005	1	⬢	2	□	1	⬢	2	□	9	2	□	15	1	1

Table A.7

The best parameter’s configurations on the `Complex2D` datasets considering the CVI protocol with the DNs index. The symbol □ is used for a rectangular SOM grid (g) and ⬢ for a hexagonal grid.

`Complex2D`	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	res	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	kNN	S	g	kNN	S	σ
`K2_D50_L0`	1	⬢	0	0.9	0.05	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	□	1	□	2	□	1	2	□	2	0.5	2
`K3_D50_L0`	2	□	0	0.9	0.25	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	⬢	0.5	□	2	□	1	2	□	2	2	3
`K6_D50_L0`	0.5	□	0	0.1	0.5	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	□	0.5	⬢	2	□	1	2	□	2	1	1
`K8_D50_L0`	2	□	0	0.1	0.5	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	⬢	1	□	2	□	1	2	□	2	0.5	2.5
`K2_D30_L0`	0.5	□	0	0.1	0.5	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	□	1	□	2	□	1	2	□	2	2	1.5
`K3_D30_L0`	0.5	⬢	0	0.1	0.01	2	□	5.0E-4	0.01	1	□	1.0E-4	0.005	2	□	2	□	1	□	2	□	1	2	□	2	2	3
`K6_D30_L0`	2	⬢	0	0.1	0.5	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	□	2	⬢	1	□	2	□	1	2	⬢	11	2	1.5
`K8_D30_L0`	0.5	⬢	0	0.1	0.25	2	□	1.0E-4	0.001	2	□	5.0E-4	0.1	2	⬢	2	□	2	□	2	□	1	2	□	2	2	1.5
`K2_D15_L1`	0.5	⬢	0.6	0.7	0.5	0.5	□	5.0E-4	0.001	0.5	□	5.0E-4	0.001	0.5	□	2	⬢	0.5	□	2	□	1	0.5	□	2	0.5	0.5
`K3_D15_L1`	1	⬢	0	0.1	0.5	2	□	5.0E-4	0.01	1	□	5.0E-4	0.01	2	□	2	□	2	□	2	□	9	2	□	11	1	1
`K6_D15_L1`	0.5	⬢	0	0.1	0.01	2	□	1.0E-4	0.005	2	□	1.0E-4	0.01	1	⬢	1	⬢	0.5	□	2	□	2	2	□	2	0.5	1
`K8_D15_L1`	2	⬢	0.6	0.9	0.25	2	□	5.0E-5	0	2	□	1.0E-4	0.001	2	⬢	1	⬢	0.5	□	2	□	15	2	⬢	11	2	2
`K2_D10_L1`	1	⬢	0	0.9	0.25	2	□	5.0E-4	0.01	2	□	1.0E-3	0.001	2	□	2	□	0.5	⬢	0.5	⬢	1	1	□	11	2	0.5
`K3_D10_L1`	1	□	0	0.1	0.5	2	□	1.0E-4	0.005	0.5	⬢	1.0E-4	0.01	0.5	⬢	2	⬢	1	□	1	□	2	1	□	3	2	1.5
`K6_D10_L1`	2	⬢	0.6	0.7	0.5	2	⬢	5.0E-5	0.005	2	⬢	5.0E-5	0.001	1	□	1	⬢	2	⬢	2	□	15	2	□	7	2	2
`K8_D10_L1`	0.5	⬢	0.6	0.7	0.25	1	⬢	5.0E-5	0	2	□	5.0E-5	0.005	2	⬢	1	□	1	⬢	1	□	7	0.5	□	1	0.5	1
`K3_D12_L2`	1	□	0	0.1	0.5	2	□	1.0E-4	0.001	2	□	1.0E-3	0.001	2	⬢	2	⬢	0.5	⬢	2	⬢	9	2	⬢	11	0.5	2
`K6_D12_L2`	1	⬢	0	0.9	0.05	2	□	1.0E-3	0.001	2	□	5.0E-4	0.1	2	□	1	⬢	2	□	2	⬢	11	2	□	3	1	0
`K8_D12_L2`	1	⬢	0.6	0.9	0.5	2	⬢	1.0E-5	0	2	□	1.0E-3	0.001	1	⬢	1	⬢	0.5	⬢	0.5	⬢	7	0.5	⬢	3	0.5	1
`K3_D10_L2`	1	⬢	0.6	0.7	0.5	1	□	5.0E-5	0.005	1	□	1.0E-4	0.005	2	□	2	□	1	□	2	□	1	2	□	2	1	0
`K6_D10_L2`	1	⬢	0	0.1	0.5	1	□	1.0E-5	0.001	2	⬢	5.0E-5	0.001	1	□	2	⬢	1	⬢	2	⬢	5	1	□	9	1	0
`K8_D10_L2`	1	□	0	0.1	0.5	2	⬢	1.0E-4	0	2	⬢	1.0E-4	0.001	2	□	2	□	0.5	□	2	□	9	2	⬢	11	0.5	1.5

Table A.8

The best parameter's configurations on the GENE datasets considering the CVI protocol with the CH index. The symbol ☐ is used for a rectangular SOM grid (g) and ⬢ for a hexagonal grid.

GENE	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	res	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	kNN	S	g	kNN	S	σ
bladderC.	0.5	⬢	0	0.1	0.5	0.5	☐	1.0E-1	0.1	0.5	⬢	5.0E-4	0.001	1	⬢	1	⬢	0.5	☐	0.5	☐	9	0.5	⬢	9	1	0
breastC.	0.5	☐	0	0.1	0.25	1	☐	1.0E-1	0.1	1	☐	1.0E-1	0.005	2	☐	1	☐	0.5	⬢	1	⬢	1	1	⬢	1	1	0
breastC.T.	2	☐	0.6	0.7	0.5	1	⬢	5.0E-4	0.01	1	⬢	5.0E-4	0.01	2	☐	1	⬢	0.5	⬢	1	⬢	1	1	☐	9	1	1
carcinomas	0.5	☐	0	0.1	0.25	1	⬢	1.0E-4	0.005	0.5	⬢	5.0E-4	0.001	2	☐	2	☐	1	⬢	2	☐	3	2	☐	15	0.5	1.5
centNrvSys.	0.5	☐	0	0.3	0.01	0.5	☐	1.0E-2	0.005	0.5	☐	1.0E-1	0.5	1	⬢	2	☐	0.5	☐	2	☐	3	2	☐	5	0.5	0.5
endometr.C.	0.5	⬢	0	0.1	0.5	2	⬢	5.0E-4	0.001	0.5	⬢	5.0E-4	0.001	2	⬢	2	☐	2	☐	2	⬢	7	1	⬢	9	1	0
glioblast.	0.5	☐	0	0.7	0.5	0.5	⬢	1.0E-3	0.01	0.5	⬢	5.0E-4	0.001	2	☐	2	☐	2	☐	0.5	☐	9	1	⬢	9	0.5	1
gliomagen.	0.5	⬢	0	0.1	0.5	0.5	☐	1.0E-3	0.001	1	⬢	5.0E-4	0.01	1	☐	1	⬢	1	☐	2	☐	7	0.5	☐	7	1	3
gliomas	1	⬢	0	0.3	0.1	2	☐	1.0E-3	0	0.5	⬢	5.0E-4	0.001	1	⬢	2	☐	0.5	⬢	0.5	⬢	1	1	☐	9	1	1
hepatocell.C.	1	☐	0	0.9	0.5	1	⬢	5.0E-4	0.01	0.5	⬢	5.0E-4	0.001	2	☐	1	⬢	0.5	⬢	1	⬢	1	2	☐	15	2	2.5
leukemia-1	2	☐	0.6	0.7	0.5	0.5	⬢	1.0E-2	0	0.5	⬢	1.0E-2	0.1	0.5	⬢	2	☐	0.5	☐	0.5	☐	9	0.5	☐	9	2	2
leukemia-2	1	⬢	0	0.1	0.25	1	⬢	5.0E-4	0.01	1	⬢	5.0E-4	0.01	1	⬢	2	☐	0.5	☐	1	⬢	5	0.5	☐	5	0.5	1
leukemia-3	0.5	☐	0	0.7	0.5	0.5	⬢	1.0E-3	0.01	0.5	⬢	1.0E-2	0.01	0.5	⬢	2	☐	2	☐	1	⬢	1	1	⬢	9	2	1.5
lungT.-1	0.5	⬢	0	0.1	0.5	0.5	⬢	1.0E-4	0.005	2	☐	5.0E-4	0.01	2	☐	1	⬢	1	⬢	1	⬢	1	1	⬢	9	1	1
lungT.-2	0.5	⬢	0	0.5	0.5	2	☐	1.0E-3	0	0.5	⬢	2.0E+0	0.1	2	⬢	2	☐	0.5	⬢	1	☐	1	1	☐	9	1	1
lymphoma-1	1	☐	0	0.1	0.5	1	⬢	1.0E-1	0.001	1	⬢	1.0E+0	0.05	1	⬢	1	⬢	2	⬢	1	⬢	5	1	☐	9	1	0
lymphoma-2	0.5	⬢	0.4	0.5	0.1	2	☐	5.0E-4	0.01	2	☐	5.0E-4	0.01	2	☐	1	⬢	0.5	⬢	2	☐	9	2	☐	15	0.5	0.5
melanoma	2	☐	0	0.1	0.5	0.5	☐	1.0E-3	0.001	0.5	☐	1.0E-2	0.001	1	⬢	2	☐	1	⬢	1	⬢	1	0.5	☐	5	0.5	1
mesothel.	1	☐	0.6	0.7	0.5	1	⬢	1.0E-2	0	1	⬢	1.0E-2	0.005	1	⬢	2	☐	0.5	☐	1	⬢	1	1	☐	1	1	1
multitissue	2	⬢	0	0.3	0.5	1	☐	1.0E-4	0.1	0.5	⬢	5.0E-4	0.001	2	☐	1	☐	0.5	☐	2	☐	1	0.5	⬢	1	2	0.5
prostateC.-1	0.5	⬢	0	0.1	0.5	1	⬢	5.0E-4	0.01	0.5	⬢	5.0E-4	0.001	2	☐	2	⬢	0.5	⬢	1	⬢	1	0.5	☐	5	2	3
prostateC.-2	1	⬢	0	0.1	0.25	0.5	☐	1.0E-2	0.5	0.5	⬢	5.0E-4	0.001	2	☐	2	☐	1	⬢	2	☐	3	1	⬢	9	1	3
prostateC.-3	0.5	⬢	0	0.1	0.5	0.5	⬢	1.0E-3	0.01	0.5	⬢	5.0E-4	0.001	0.5	⬢	2	☐	0.5	☐	0.5	⬢	1	0.5	☐	7	2	3
roundBluecellT.	1	⬢	0	0.1	0.5	0.5	☐	5.0E-4	0.01	0.5	☐	5.0E-4	0.01	0.5	⬢	2	☐	2	⬢	0.5	⬢	1	0.5	⬢	9	2	2
serratedC.	2	⬢	0	0.1	0.5	1	☐	1.0E+0	0.01	0.5	☐	1.0E-1	0.001	2	⬢	1	☐	0.5	☐	1	☐	1	1	☐	9	1	3

Table A.9

The best parameter's configurations on the REAL datasets considering the CVI protocol with the CH index. The symbol □ is used for a rectangular SOM grid (g) and ⬢ for a hexagonal grid.

REAL	Clusot*					gSOM				gSOM*				SOMKm		SOMKm*		SOMNcut		SOMSpec			SOMSpec*			SOMStar*	
dataset	S	g	θ_0	θ	res	S	g	G	ΔG	S	g	G	ΔG	S	g	S	g	S	g	S	g	kNN	S	g	kNN	S	σ
breastC.	1	⬢	0	0.1	0.5	0.5	⬢	1.0E-3	0.05	0.5	⬢	1.0E-3	0.01	1	□	2	□	1	⬢	2	⬢	2	0.5	□	5	0.5	1.5
contrac.	0.5	⬢	0	0.3	0.25	0.5	⬢	5.0E-4	0.005	0.5	⬢	5.0E-4	0.005	1	⬢	1	□	1	⬢	1	⬢	3	1	⬢	7	2	2
fertility	0.5	⬢	0	0.1	0.05	0.5	⬢	5.0E-4	0.01	0.5	⬢	5.0E-4	0.005	0.5	⬢	2	□	0.5	⬢	0.5	⬢	1	0.5	⬢	7	0.5	0.5
glass	0.5	⬢	0	0.1	0.05	2	⬢	5.0E-5	0.01	0.5	⬢	5.0E-4	0.001	2	⬢	2	⬢	1	□	2	□	15	2	□	5	0.5	1
ionosph.	0.5	⬢	0	0.1	0.05	0.5	⬢	5.0E-4	0	0.5	⬢	5.0E-4	0.01	0.5	⬢	2	□	0.5	⬢	0.5	⬢	1	0.5	⬢	7	1	1
iris	2	⬢	0	0.1	0.05	1	⬢	5.0E-4	0.005	0.5	□	1.0E-3	0.05	2	□	1	⬢	0.5	⬢	0.5	□	9	2	□	1	0.5	3
laryngeal3	1	⬢	0	0.3	0.05	0.5	⬢	1.0E-4	0	0.5	□	1.0E-3	0.05	2	⬢	2	□	0.5	⬢	0.5	⬢	1	0.5	⬢	1	2	0
letterABC	0.5	□	0	0.3	0.25	0.5	□	1.0E-4	0	0.5	□	1.0E-4	0.001	2	⬢	2	⬢	1	⬢	1	⬢	2	2	□	1	0.5	2.5
respirat.	0.5	□	0	0.3	0.25	0.5	⬢	5.0E-4	0.005	0.5	⬢	5.0E-4	0.005	2	⬢	2	⬢	0.5	⬢	0.5	⬢	1	0.5	⬢	7	0.5	1
segment.	1	⬢	0	0.1	0.5	2	□	1.0E-3	0.5	0.5	□	1.0E-3	0.001	2	□	2	□	0.5	⬢	0.5	⬢	1	0.5	⬢	7	2	2.5
voice_3	0.5	⬢	0	0.1	0.05	0.5	□	1.0E-3	0.01	0.5	□	5.0E-4	0.05	2	□	2	□	0.5	⬢	1	⬢	1	0.5	□	5	0.5	1.5
weaning	2	⬢	0	0.1	0.05	0.5	□	1.0E-4	0	0.5	□	1.0E-3	0.05	0.5	⬢	2	□	0.5	⬢	0.5	⬢	3	0.5	⬢	7	0.5	1.5
wine	1	□	0	0.1	0.5	0.5	⬢	5.0E-4	0.005	0.5	⬢	5.0E-4	0.005	2	□	2	□	1	⬢	1	⬢	1	1	⬢	11	2	2
wineRed	0.5	□	0	0.3	0.25	0.5	⬢	5.0E-4	0.005	0.5	⬢	1.0E-3	0.005	2	□	2	□	0.5	⬢	0.5	⬢	11	2	□	2	0.5	2
yeast	0.5	⬢	0	0.1	0.05	0.5	□	5.0E-5	0	0.5	⬢	5.0E-5	0.005	2	□	2	□	0.5	⬢	2	⬢	2	1	□	7	0.5	1.5

Supplement to chapter “Weighted Cluster Ensemble”

B

B.1 Comparison of single-clustering algorithms

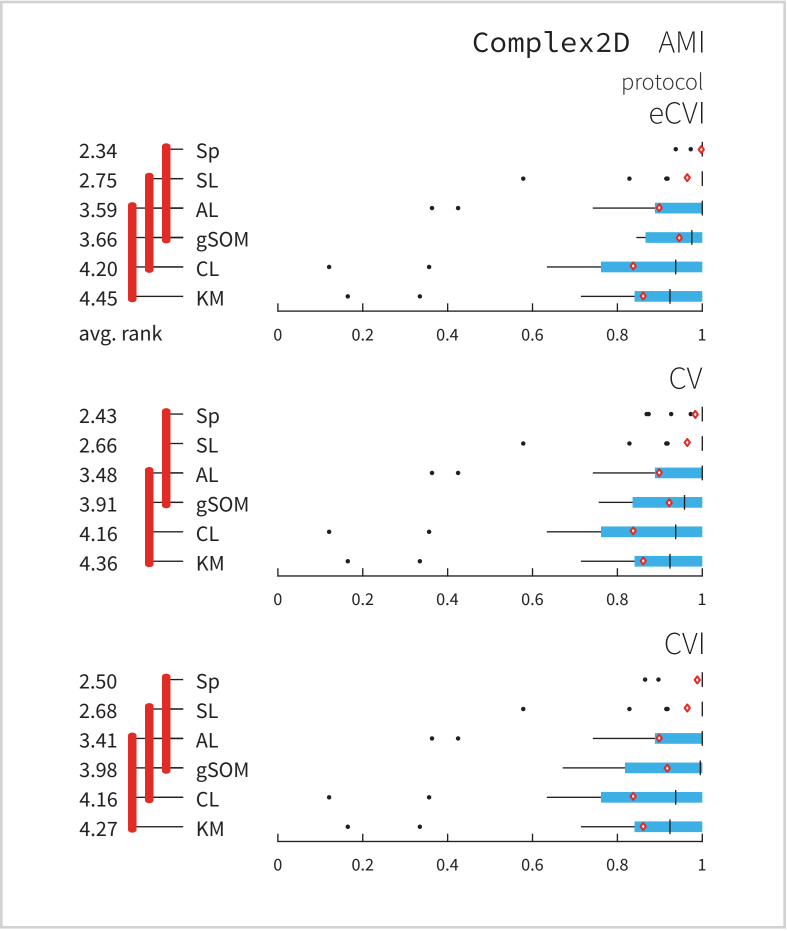


Figure B.1

Average ranks of algorithms based on the AMI score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

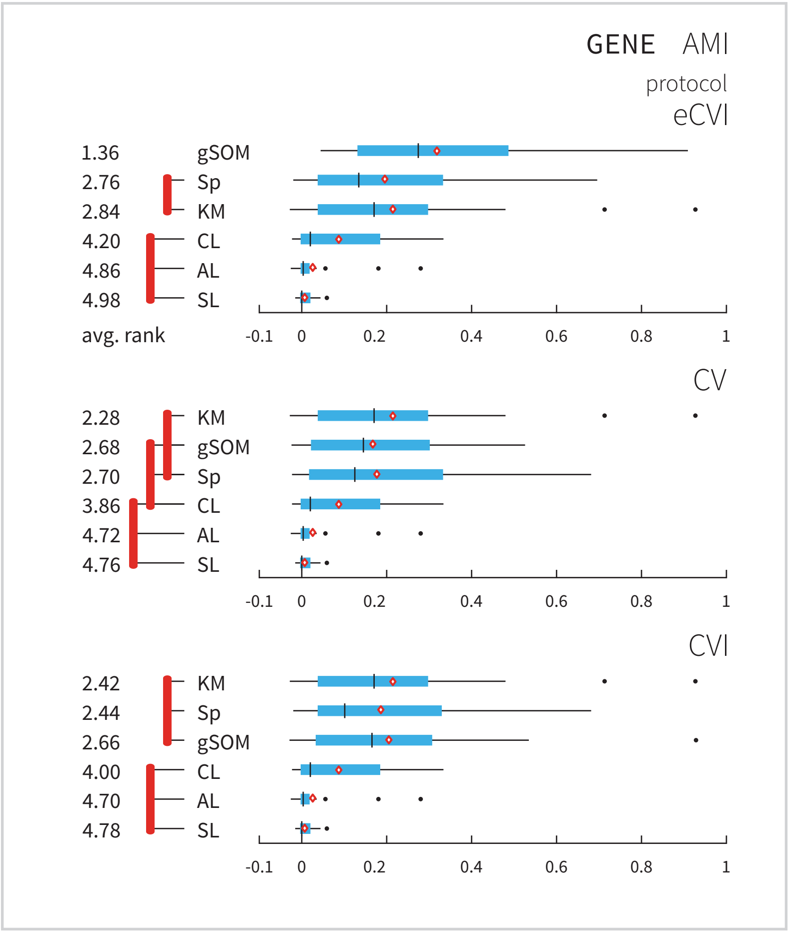


Figure B.2

Average ranks of algorithms based on the AMI score across all the GENE datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

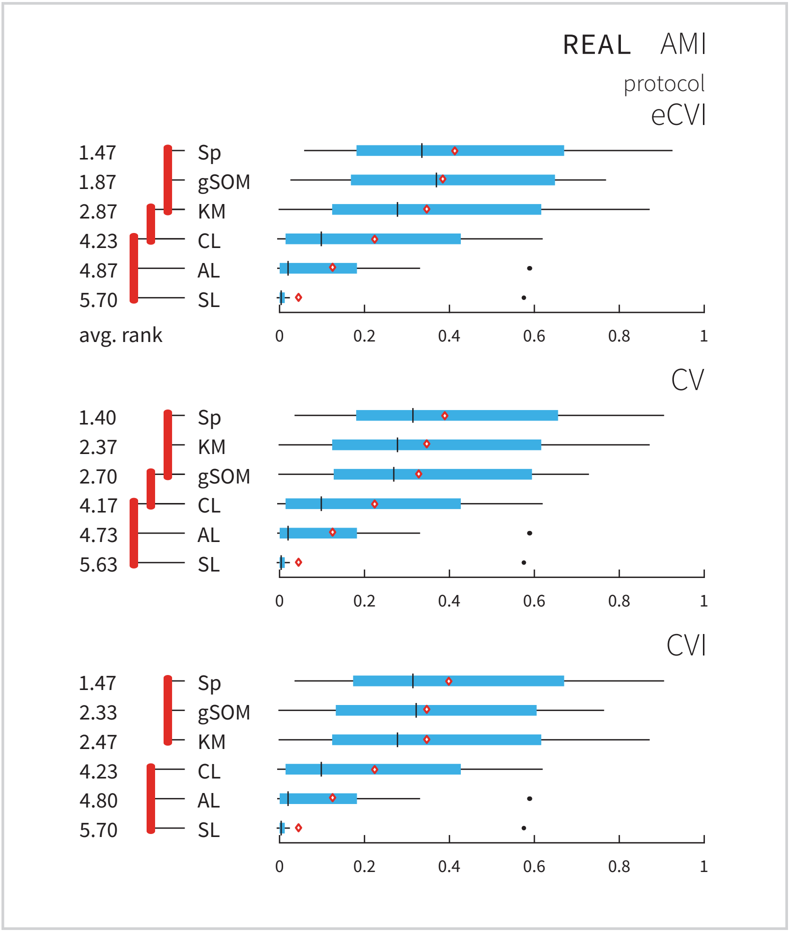


Figure B.3

Average ranks of algorithms based on the AMI score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

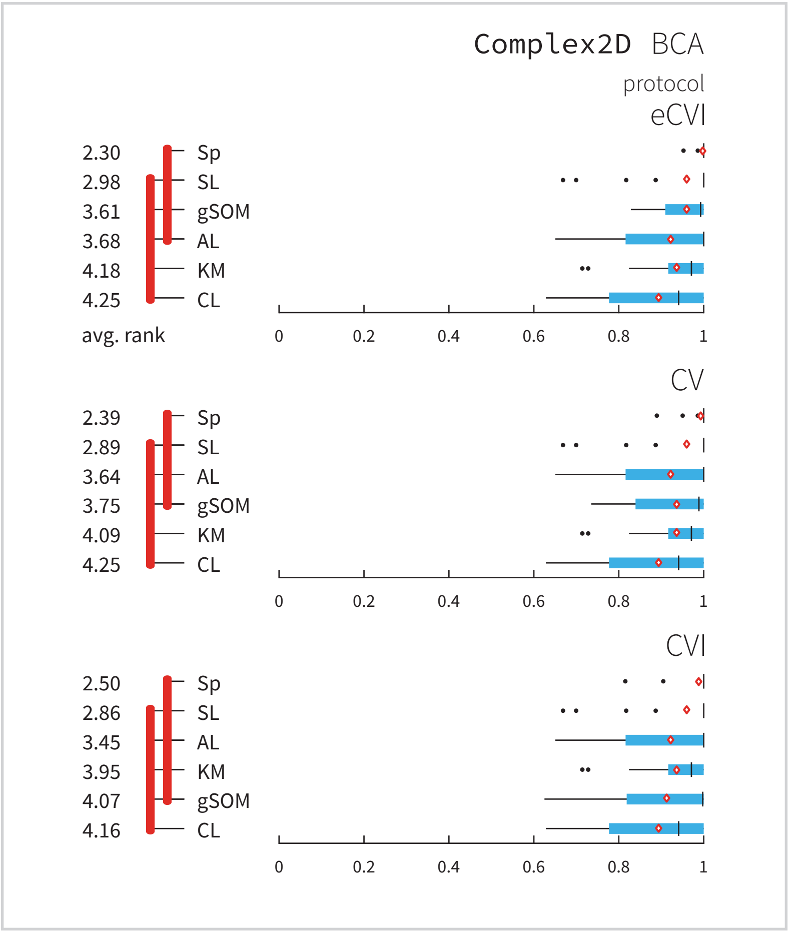


Figure B.4

Average ranks of algorithms based on the BCA score across all the `Complex2D` datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

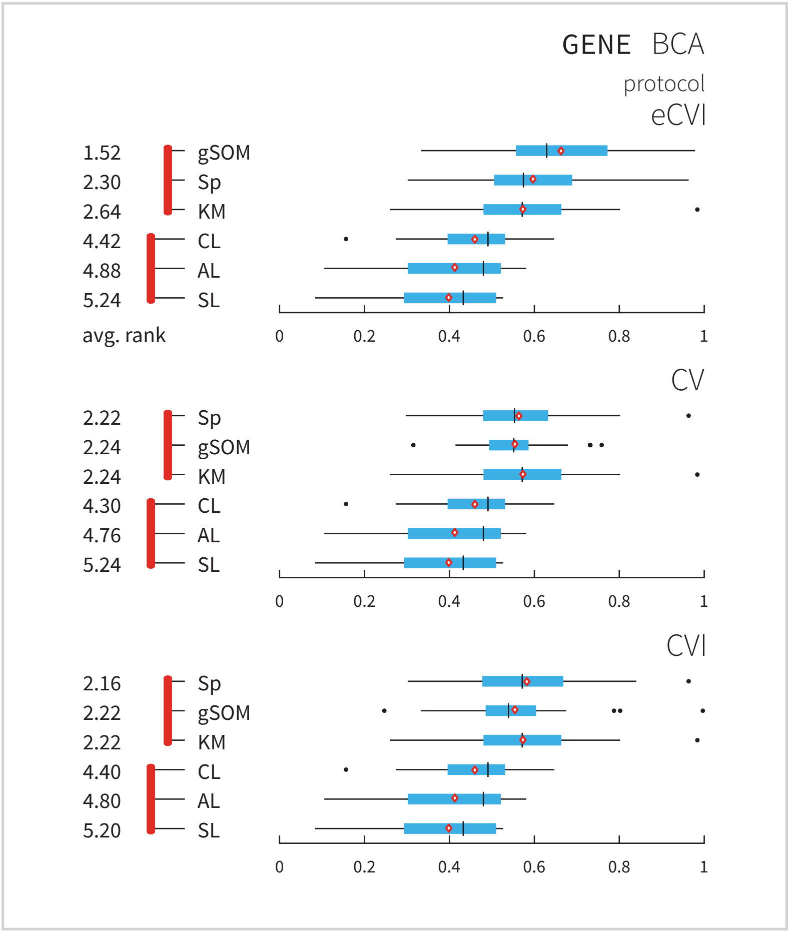


Figure B.5

Average ranks of algorithms based on the BCA score across all the GENE datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

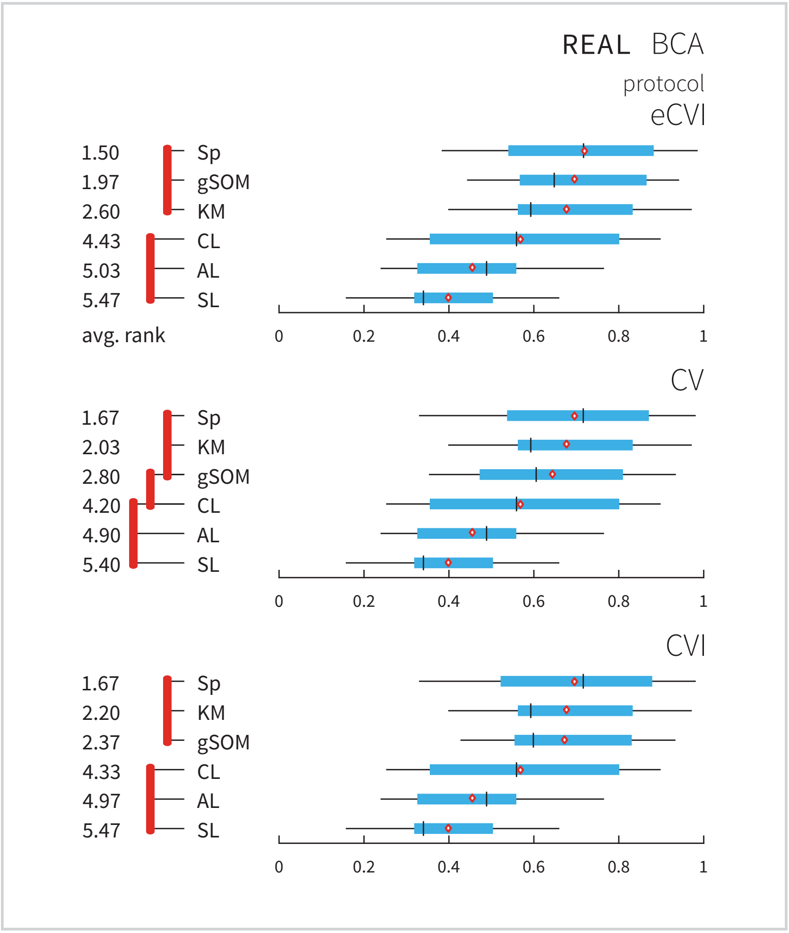


Figure B.6

Average ranks of algorithms based on the BCA score across all the REAL datasets are displayed on the left. Red lines connect algorithms with no significant difference in performance. We used Friedman's test and Bergmann-Hommel's post hoc procedure for multiple comparisons with $\alpha = 0.05$. Distribution of scores under the eCVI, CV, and CVI protocols are on the right. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

B.2 Comparison of consensus functions

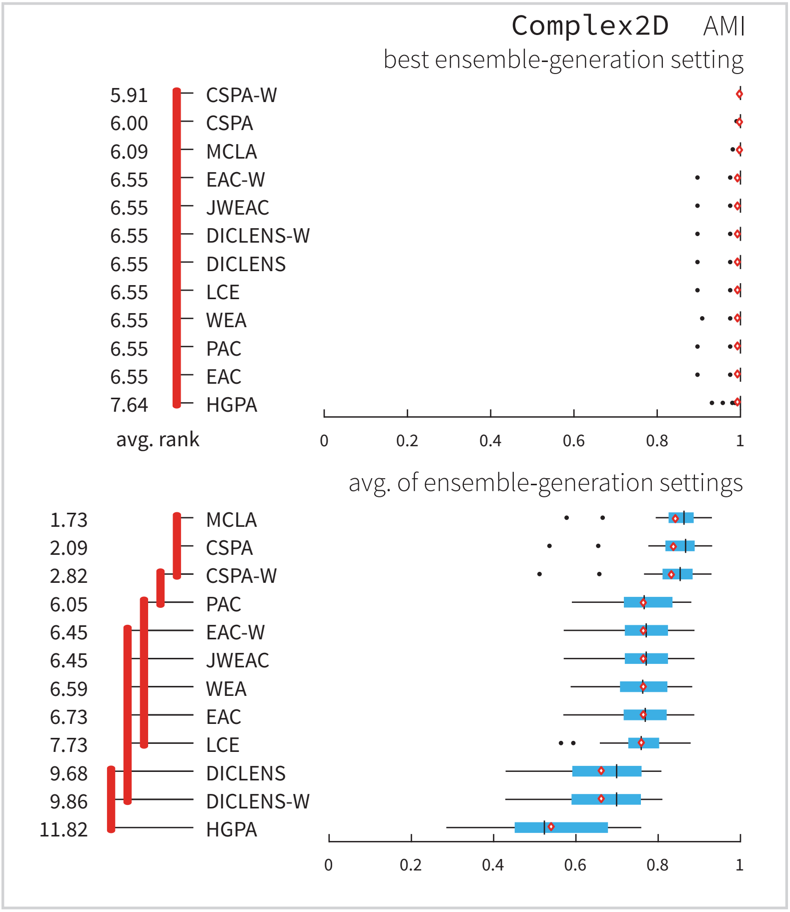


Figure B.7

Average ranks of consensus functions based on the AMI score across all the `Complex2D` datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

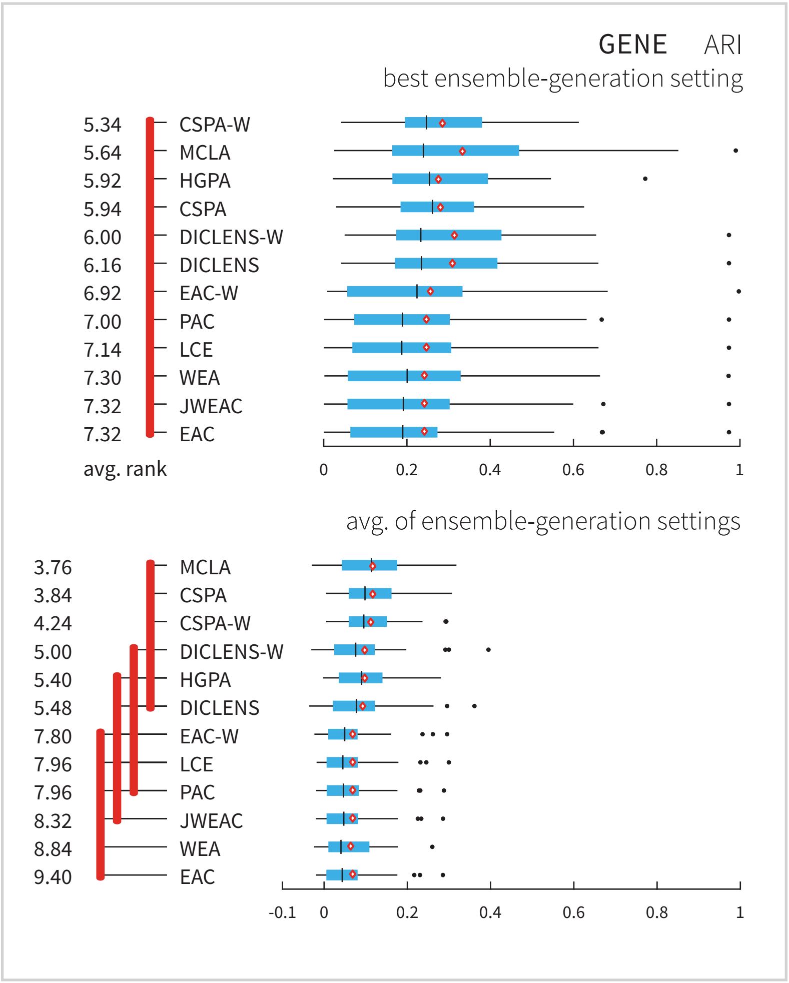


Figure B.8

Average ranks of consensus functions based on the AMI score across all the GENE datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

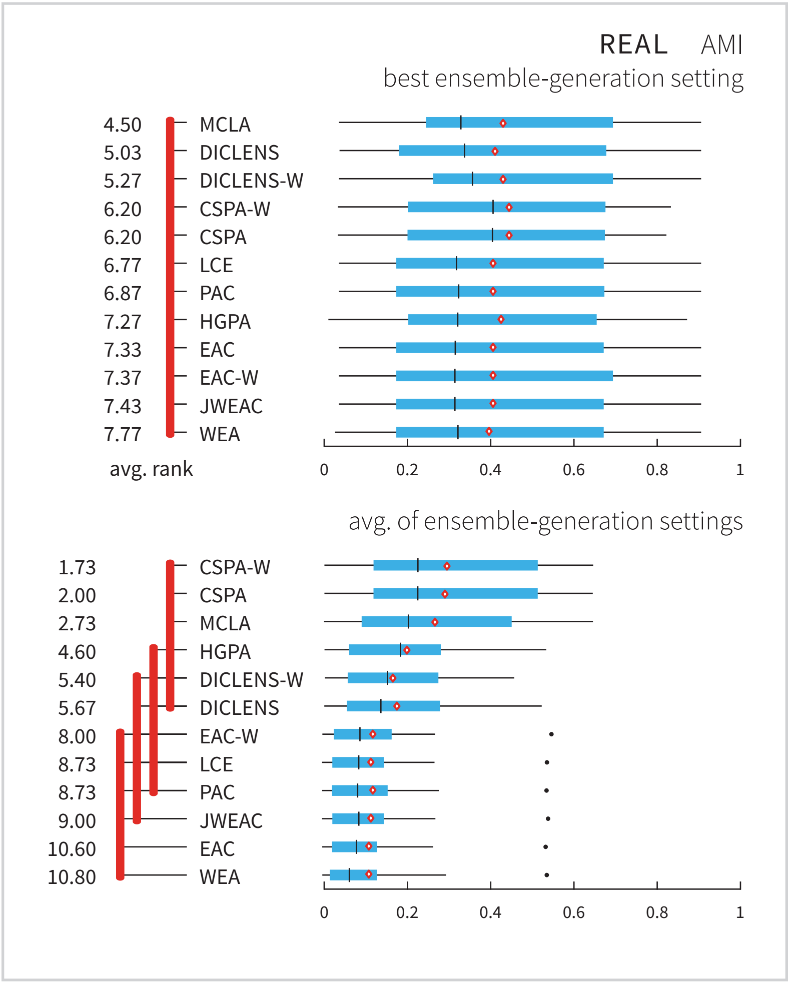


Figure B.9

Average ranks of consensus functions based on the AMI score across all the REAL datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

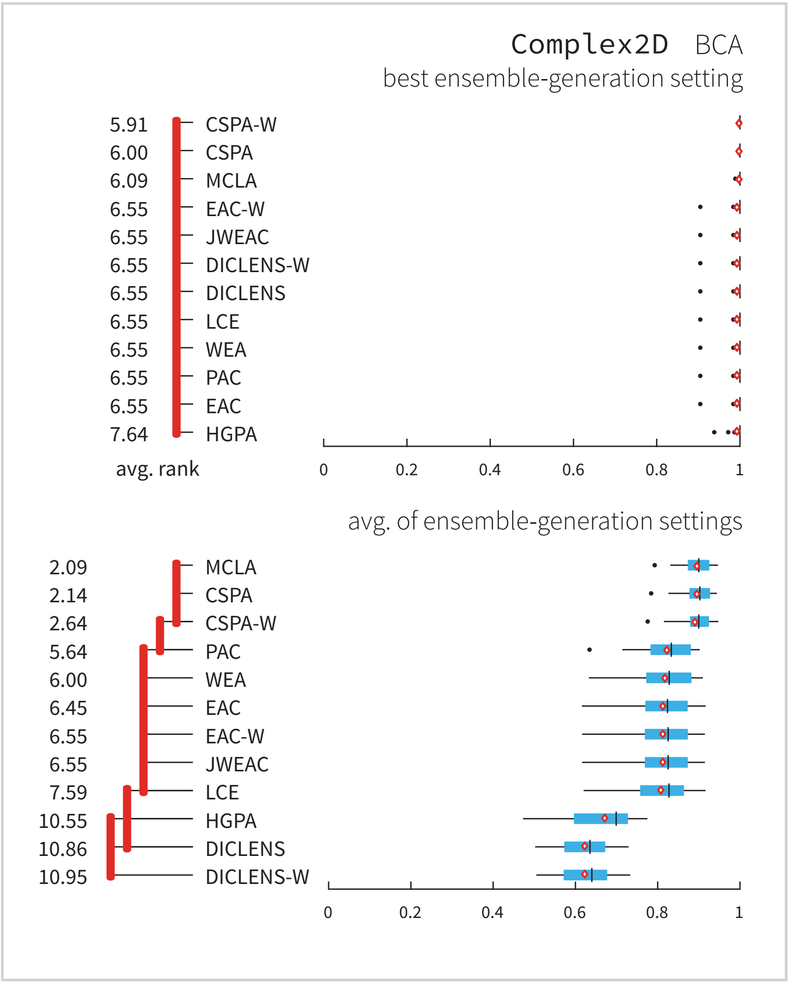


Figure B.10

Average ranks of consensus functions based on the BCA score across all the `Complex2D` datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

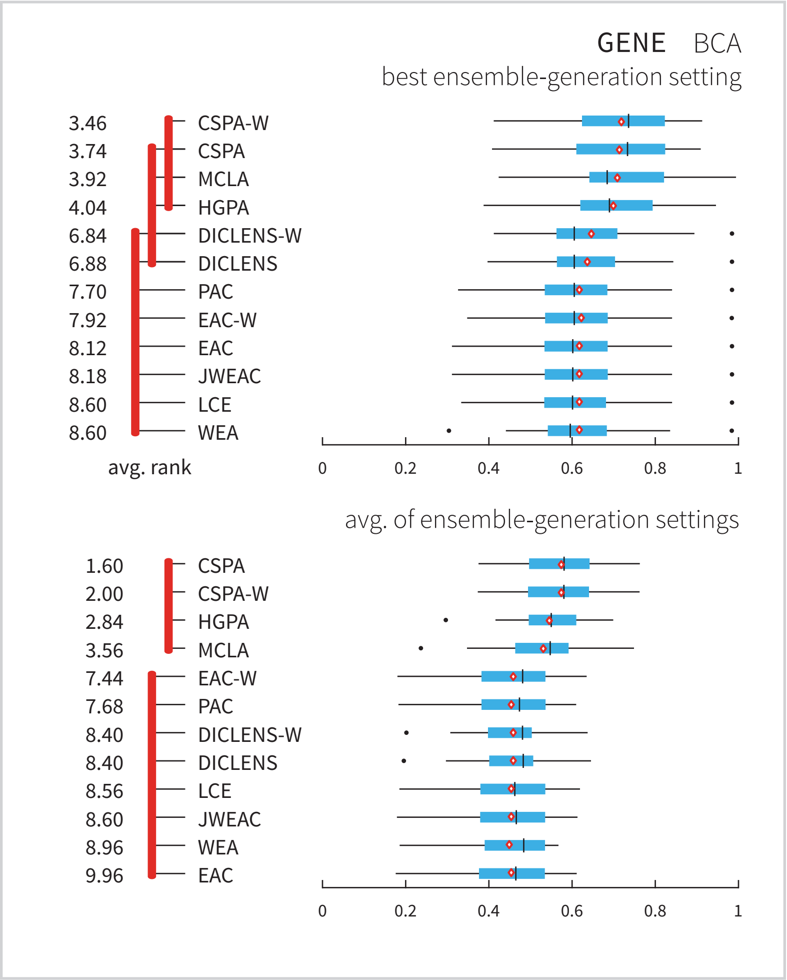


Figure B.11

Average ranks of consensus functions based on the BCA score across all the GENE datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

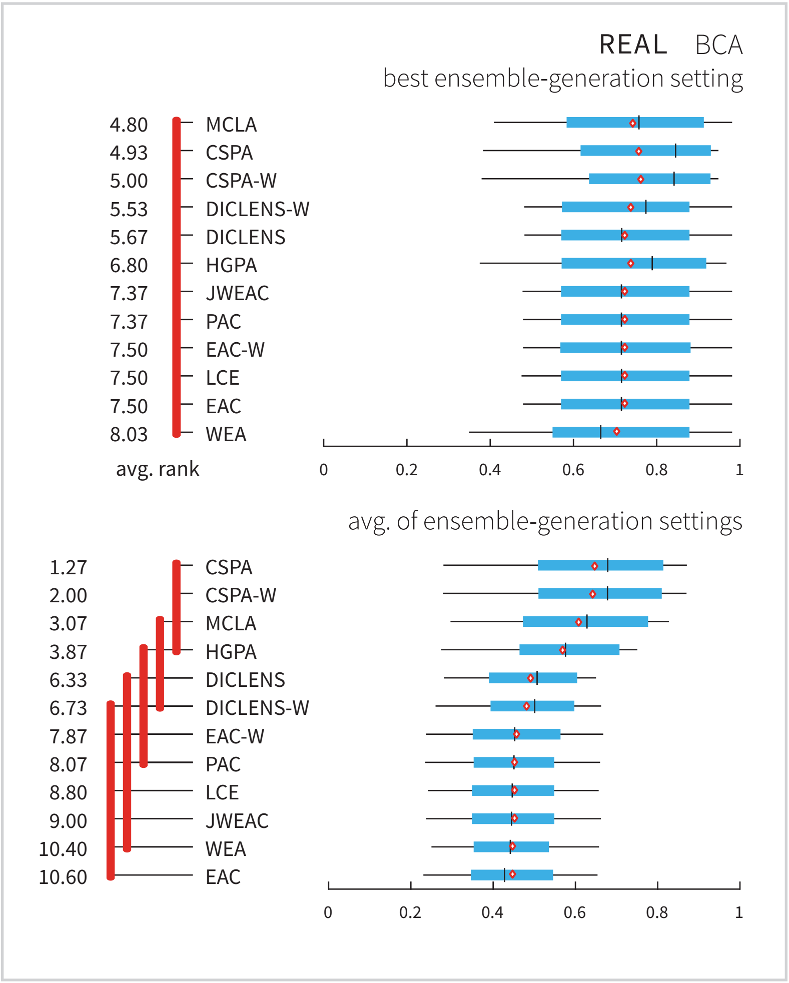


Figure B.12

Average ranks of consensus functions based on the BCA score across all the REAL datasets along with distribution of the scores are displayed. Top panel corresponds to the best ensemble-generation setting and the bottom to the average of all settings. Red lines connect algorithms with no significant difference in performance using Friedman's test and Shaffer's post hoc procedure for multiple comparisons with $\alpha = 0.05$. The median score is marked with a short black line and the average score with a red diamond. The edges of blue rectangles indicate the first and third quartiles, and whiskers extend to the most extreme scores within 1.5 times the interquartile range. Scores beyond the whiskers are outliers and are displayed as black dots. The algorithms are ordered by the average ranks.

B.3 Stan code for modelling win probabilities

```
data {
  int n;              // number of samples
  int m;              // number of competing methods
  int isWinner[n,m]; // indicator if competing alternative was
  real nWinners[n];  // row sums of isWinner (1 = single winner, 2+ = tied)
  vector[m] prior;   // (typically, set to 1/2)
}

parameters {
  simplex[m] B;      // win probabilities
}

model {
  B ~ dirichlet(prior);

  for (i in 1:n)
    for (j in 1:m)
      if (isWinner[i,j] == 1)
        increment_log_prob(log(1/nWinners[i]) + categorical_log(j, B));
}
```

Razširjeni povzetek

C.1 Uvod

Naša življenja so preobložena s podatki vseh vrst, ki jih želimo interpretirati in iz njih pridobiti razumevanje in znanje. Razvrščanje podatkov v gruče je temeljno orodje na področju strojnega učenja, razpoznave vzorcev [1] in rudarjenja podatkov [2, 3], kjer zaradi nenehnega povečevanja količine podatkov narašča tudi potreba po njihovi učinkoviti analizi, vizualizaciji in interpretaciji. Razvrščanje v gruče ima dolgo in bogato zgodovino, ki se je intenzivneje začela pred 60 leti [4] in zanimanje za to disciplino vsekakor ne ugaša – razumljivo, saj se z vsakim izumom na področju informacijsko-komunikacijske tehnologije poveča tudi potreba po samodejni obdelavi zajetih podatkov. Raziskovalci strojnega učenja imajo v svoji orodjarni na ducate algoritmov za analizo podatkov in navadno je algoritem za razvrščanje prvi na vrsti, ko gre za spoznavanje z novimi podatki. Težko je našteti vsa področja uporabe algoritmov za razvrščanje, navedimo nekaj najpomembnejših: razčlenjevanje slik, razvrščanje besedil, analiza izražanja genov, oddaljeno zaznavanje v geologiji, taksonomska analiza v biologiji, analiza trga, patološke raziskave in diagnostika ter preučevanje socioloških pojavov [5].

Razvrščanje je postopek, kjer podatke ali vzorce razvrstimo v gruče na podlagi medsebojne podobnosti. Tisti podatki, ki so si na nek način podobni, naj spadajo v isto gručo, medtem ko naj si bodo podatki v različnih gručah čim bolj različni [6]. Metode za razvrščanje spadajo med metode za nenadzorovano učenje, kjer vnaprejšnje znanje o podatkih ni podano, kar pomeni, da ne poznamo primerov že uvrščenih ali označenih podatkov in tudi ni vnaprej definiranih gruč ali razredov [7]. To je pomembna razlika, ki ločuje razvrščanje od uvrščanja oziroma klasifikacije. Slednje deluje kot nadzorovano učenje, kjer sta dana število razredov in razvrstitev določenega dela podatkov. Kleinberg je z *izrekom nemogočega* pokazal, da idealnega algoritma za razvrščanje ni [8], zato se srečujemo s konstantno rastjo števila novih pristopov in njihovih različic.

Prav zaradi univerzalnosti problema razvrščanja na eni strani in prej omenjenega izreka nemogočega, ki zanika obstoj splošne optimalne metode za razvrščanje, na drugi strani, se izkaže, da je zelo dobrodošla analiza danih podatkov z različnimi metodami. Že star slovenski pregovor pravi, da več glav več ve. V splošnem tudi raje zaupamo raznoliki komisiji strokovnjakov kot posamezniku. Točno to je tudi glavna ideja pristopa z ansambli [9], kjer želimo na isti problem pogledati z različnih zornih kotov. Pri tem moramo rešiti problem združevanja različnih pogledov v enoten rezultat, ki mu pravimo tudi sporazumna razvrstitev. Algoritem, ki iz množice razvrstitev oziroma ansambla izračuna sporazumno rešitev, imenujemo združevalna funkcija. Celotnemu pristopu pa pravimo razvrščanje z uporabo ansamblov ali metode ansamblov [10].

Zdaj si predstavljajmo, da smo nekajkrat razvrstili dane podatke in da bi iz te množice razvrstitev ali ansambla radi dobili sporazumno razvrstitev. Postavi se vprašanje: ali so vse razvrstitve v ansamblu enako *dobre*? Če niso, zakaj bi potem morale vse imeti enako težo? Tu nastopijo metode posebne discipline na področju razvrščanja, imenovane ocenjevanje razvrstitev [11]. Kazalec veljavnosti razvrstitve (CVI) je mera, ki določa *kakovost* določene razvrstitve [12]. V osnovi ločimo med notranjimi in zunanjimi kazalci. Slednji izračunajo ujemanje med dano razvrstitvijo in referenčno razvrstitvijo, ki jo v tem kontekstu razumemo kot pravilno ali ciljno in je podana s strani poznavalcev področja. Tako so zunanji kazalci primerni za objektivno merjenje uspešnosti algoritmov za razvrščanje in njihovo primerjavo. Notranji kazalci veljavnosti nimajo dostopa do drugih informacij razen do podatkovne množice in razvrstitve, ki jo je napravil algoritem. Zato notranje kazalce uporabljamo, ko pravilna razvrstitev ni znana vnaprej in to je najpogostejši scenarij. Potemtakem bi bila morda dobra ideja, da ocenimo vsako razvrstitev v ansamblu z notranjimi kazalci veljavnosti, preden uporabimo združevalno funkcijo. Tako lahko uporabimo ocene, da z njimi utežimo razvrstitve v ansamblu in jih nato ustrezno združimo. To je kratek povzetek postopka, ki se imenuje analiza pomembnosti razvrstitev (ang. partition relevance analysis, PRA), ki so ga prvi uporabili Duarte idr. [13]. Predstavili so postopek, kjer več notranjih kazalcev oceni ansambel razvrstitev. Nato se iz njihovega povprečja izračunajo uteži in se uporabijo v združevalni funkciji. Pristop, kjer pri združevanju ansambla razvrstitev uporabimo uteži, imenujemo *razvrščanje z metodo uteženih ansamblov*. V tem kontekstu smo v disertaciji obravnavali naslednje tri korake:

1. Gradnja ansambla razvrstitev z enostavnimi[1] algoritmi za razvrščanje.
2. Uteževanje razvrstitev v ansamblu z uporabo notranjih kazalcev veljavnosti.
3. Upoštevanje uteži v procesu združevanja ansambla v sporazumno razvrstitev.

[1]Besedno zvezo *enostavni algoritmi za razvrščanje* (ang. single-clustering algorithms) uporabljamo zgolj v kontekstu ansamblov, da ločimo med metodami, ki izdelajo posamezno razvrstitev in tistimi, ki te razvrstitve združijo v sporazumno rešitev – to so združevalne funkcije.

C.1.1 Prispevki k znanosti

Glavna pozornost pričujoče disertacije je usmerjena k izboljšavam obstoječih pristopov razvrščanja podatkov v gruče na ravni analize z enostavnimi algoritmi, ocenjevanja rešitev z notranjimi kazalci veljavnosti in iskanja sporazumne razvrstitve z uporabo metode ansamblov. Tu povzemamo svoje izvirne prispevke k znanosti.

- *Razvoj novega algoritma za razvrščanje, ki povezuje samo-organizirajočo nevronsko mrežo s principi gravitacijskega razvrščanja.* Predlagamo algoritem gSOM, ki opravi razvrščanje podatkov na dvostopenjski način [15]. Na prvi stopnji abstrahira podatke s Kohonenovo samo-organizirajočo mrežo (SOM) [16]. Nato se na drugi stopnji nevroni mreže SOM združujejo po principu gravitacijske sile [17]. Število gruč določi gSOM samodejno. Rezultati preizkusov na umetnih, genetskih in drugih realnih podatkih kažejo na učinkovitost predlaganega algoritma, ki ga uporabimo tudi v postopku gradnje ansambla razvrstitev [18].

- *Predlog notranjega kazalca veljavnosti razvrstitev, ki temelji na teoriji grafov, za uteževanje razvrstitev v ansamblu.* Spremenili smo dobro poznan Dunnov notranji kazalec veljavnosti [19] in razvili kazalec DNs, osnovan na najkrajših poteh v Gabrielovem grafu nad podatki [20]. Iz strokovne literature smo izbrali 33 kazalcev in jih primerjali s kazalcem DNs na umetnih in realnih podatkih raznovrstnih domen. Po našem najboljšem vedenju je to prva raziskava, ki sistematično ugotavlja zmogljivosti notranjih kazalcev v primeru podatkov z linearno neločljivimi gručami. Vsi primerjani kazalci so nato vključeni v sistem za analizo pomembnosti razvrstitev v kontekstu metod uteženih ansamblov.

- *Izboljšava pristopa uteženega ansambla za razvrščanje z uporabo analize pomembnosti razvrstitev z dodatnim korakom redukcije.* Dve glavni vprašanji pri zasnovi sistema za analizo pomembnosti razvrstitev v ansamblu (PRA) sta: koliko in katere kazalce veljavnosti razvrstitev izbrati? Ti vprašanji naslovimo s predlogom dodatnega koraka redukcije (PRAr), ki zmanjša nabor kazalcev s postopki izbire in izločanja značilnic. Učinkovitost predloga PRAr merimo v delovanju s tremi znanimi združevalnimi funkcijami in to primerjamo v obširni študiji z več različnimi načini gradnje ansamblov.

- *Generator umetnih podatkov z nadzorom linearne ločljivosti med gručami.* Za primerjavo in ovrednotenje različnih stopenj postopka razvrščanja v gruče smo razvili generator dvodimenzionalnih podatkov. Sposoben je ustvariti gruče zapletenih oblik in nadzorovati najmanjšo razdaljo med njimi skupaj s stopnjo linearne ločljivosti. Ustvarili smo družino podatkovnih množic, `Complex2D`, in jo uporabili v empirični primerjavi enostavnih algoritmov za razvrščanje, kazalcev veljavnosti razvrščanja in metod ansamblov.

V sklopu doktorske disertacije smo razvili paket orodij za MATLAB, imenovan Pepelka. Dostopen je na naslovu `http://laspp.fri.uni-lj.si/nejci/Pepelka`.

C.2 *Ustvarjanje umetnih podatkov*

Glavni namen raziskovalcev na področju strojnega učenja in računalniške znanosti v splošnem je reševanje problemov iz resničnega sveta. Razvijajo algoritme za obdelavo "realnih" podatkov – to so v osnovi zabeležene vrednosti izbranih značilnic nekega problema ali pojava, ki ga naslavljamo. Zgodi pa se, da med analizo podatkov odkrijemo lastnosti podatkov in njihovo strukturo, vendar pa s tem ne moremo manipulirati. To je razlog, zakaj si nekateri želijo ustvariti svoje lastne podatke na nadzorovan način. Ta postopek imenujemo *ustvarjanje umetnih podatkov* in orodju *generator umetnih podatkov*. Še posebej je to uporabno za preverjanje in razvijanje specifičnih lastnosti algoritmov za učenje.

Avtorji *Temeljne zbirke problemov za razvrščanje* [21] so kot enega ključnih izzivov na področju razvrščanja izpostavili gruče, ki niso ločljive s hiperravnino – za njih pravimo, da niso linearno ločljive. Z drugimi besedami: vzorci podatkov ene gruče ležijo delno ali popolnoma v konveksni ovojnici druge gruče. Po našem vedenju še ni bil razvit generator podatkov, ki bi bil sposoben nadzirati stopnjo linearne ločljivosti med gručami. Naš prispevek je ravno tovrsten generator, ki omogoča sistematično preučevanje in primerjavo zmogljivosti različnih algoritmov za razvrščanje. Razvili smo generator dvorazsežnih podatkov, ki so razporejeni v gruče raznolikih oblik. Uporabnik lahko določa število gruč, število vzorcev v vsaki gruči, obliko gruče, porazdelitev vzorcev znotraj gruče, najmanjšo razdaljo med gručami in stopnjo linearne ločljivosti med gručami, tj. koliko parov gruč je ločljivih s premico oziroma hiperravnino v splošnem primeru. Za ugotavljanje linearne ločljivosti dveh gruč uporabimo linearno programiranje s simpleksno metodo. Iščemo tako hiperravnino, ki ima vse vzorce ene gruče na eni strani in vse vzorce druge gruče na drugi strani. Če tako hiperravnino najdemo, pravimo, da sta gruči linearno ločljivi in če ne, da sta linearno neločljivi.

Če želimo vplivati na najmanjšo razdaljo med gručami in na stopnjo linearne ločljivosti, moramo definirati "telo" oziroma zunanje meje gruče. Za to uporabimo zasnovo α-oblik [31, 32] kot pripomoček za opis oblike ali obrisa množice podatkov na ravnini. α-oblika je posplošitev pojma konveksne ovojnice, kjer je α parameter, ki določa raven podrobnosti oblike. Definicija α-oblike je sledeča: če na krožnici s polmerom α ležita natanko dve točki, potem med njima obstaja povezava ali rob. α-oblika je množica vseh takih povezav.

Predlagan generator lahko ustvari oblike gruč z različnimi stopnjami kompleksnosti: strnjene in kroglaste; podolgovate; s koti in luknjami, ki lahko vsebujejo druge gruče. Ustvarili smo družino podatkovnih množic, ki smo jo poimenovali `Complex2D`. Namenjena je odkrivanju vpliva razdalje med gručami in njihove nelinearne prepletenosti na uspešnost algoritmov za razvrščanje in tudi kazalcev veljavnosti razvrstitev.

C.3 Ocenjevanje razvrstitev

Predstavljenih je bilo že veliko kazalcev veljavnosti razvrstitev [11, 36–38]. Kazalce veljavnosti delimo na notranje in zunanje. Notranji kazalci (CVI) ocenijo dano razvrstitev z merjenjem strnjenosti vzorcev znotraj gruč in ločenosti med gručami, brez dodatnih informacij o referenčni razvrstitvi. Slednje pa za svoje delovanje potrebujejo zunanji kazalci (eCVI) in kot taki ponujajo objektivnejše vrednotenje. Kakorkoli že, v realnih situacijah je referenčna ali resnična razvrstitev bolj želja kot dejstvo, zato v večini primerov uporabljamo notranje kazalce, ki nam povedo, kako dobro se neka razvrstitev prilega podatkom.

Eden izmed bolj znanih notranjih kazalcev je Dunnov kazalec iz leta 1973 [19]. Strnjenost gruče ali njen premer definira kot največjo evklidsko razdaljo med dvema vzorcema znotraj iste gruče. Ločenost para gruč pa je najmanjša evklidska razdalja med dvema vzorcema, pri čemer je eden iz prve gruče in eden iz druge. Vrednost Dunnovega kazalca je razmerje med največjim premerom med vsemi gručami in najmanjšo razdaljo med vsemi pari gruč. Do danes je bilo predstavljenih že nekaj posplošitev Dunnovega kazalca, ki na drugačen način definirajo mere strnjenosti znotraj in ločitve med gručami [39, 40]. V disertaciji predstavljamo novo izpeljanko, ki temelji na posplošenem kazalcu Pala in Biswasa [39] z željo, da bi še bolje ocenjeval razvrstitve s poljubnimi oblikami gruč, kar še vedno predstavlja svojevrsten izziv na področju [41]. V preteklosti so se s tem izzivom že soočali, in sicer z uporabo grafov sosednosti med podatki, ki naj bi učinkoviteje in natančneje opisali obliko gruč [39, 42]. Pal in Biswas sta opravila poskuse s tremi vrstami grafov: minimalno vpeto drevo, graf relativne sosednosti in Gabrielov graf [43]. Njuni rezultati na raznolikih podatkih kažejo, da je kazalec, osnovan na Gabrielovem grafu, pokazal največjo uspešnost. Tudi druge raziskave so pokazale, da so lastnosti povezljivosti Gabrielovega grafa ugodne za opis gruč na podatkih [44]. Iz teh razlogov smo Gabrielov graf vzeli za temelj predlaganega kazalca. Gabrielov graf je graf, kjer obstaja povezava med dvema vzorcema a in b natanko takrat, ko ne obstaja tretji vzorec c, ki bi se nahajal znotraj hiperkrogle s premerom enakim razdalji med a in b in središčem na polovici daljice ab.

Bistvo predlaganega kazalca, ki ga označujemo s kratico DNs, je redefinicija razdalje med dvema vzorcema. To razdaljo namreč definiramo kot dolžino najkrajše poti med tema dvema vzorcema po povezavah Gabrielovega grafa. Premer gruče ali njeno strnjenost opišemo kot najdaljšo izmed najkrajših poti med vsemi pari vzorcev znotraj iste gruče. Ločenost med dvema gručama pa kot najkrajšo pot med vsemi pari vzorcev, pri čemer vzorca nista iz iste gruče. Torej smo izvirno Dunnovo definicijo strnjenosti in ločenosti, ki temelji na evklidski razdalji, zamenjali z dolžinami najkrajših poti v grafu. Poleg tega smo izračunu kazalca dodali še člen, ki kaznuje razvrstitve z večjim številom gruč, saj je to v praksi neželeno in tudi nepričakovano.

Opisan kazalec DNs smo primerjali s 33 drugimi kazalci na 1530 umetnih podatkovnih množicah, 25 množicah s podatki o izražanju genov in 15 drugih realnih množicah podatkov. Uporabili smo štiri metodologije, ki se medsebojno dopolnjujejo, poudarek pa smo dali računanju korelacije med ocenami notranjih in zunanjih kazalcev [100]. Večje kot je za neki notranji kazalec ujemanje z izbranim zunanjim kazalcem, bolj verodostojen je ta notranji kazalec. Celoten poskus lahko povzamemo v naslednjih korakih:

1. Iz opisanih zbirk podatkov `Complex2D`, `GENE` in `REAL` izberemo podatkovno množico $\mathbf{X}$. Določena je tudi referenčna ciljna razvrstitev $\mathbf{C}^T$.

2. Napravimo razvrstitev podatkovne množice $\mathbf{X}$ z algoritmom k-voditeljev (KM) [102] in spektralnim algoritmom (Sp) [103] v K gruč, kjer je $K = 2, \dots, K_{max}$. Zgornjo mejo števila gruč K_{max} določimo glede na število vzorcev v množici $\mathbf{X}$.

3. Vsako razvrstitev ocenimo s 34 notranjimi in 3 zunanjimi kazalci veljavnosti, pri čemer slednji kot referenco vzamejo razvrstitev $\mathbf{C}^T$. Ponovimo korake 1-3 za vse podatkovne množice.

4. Izračunamo mere uspešnosti po različnih metodologijah in njihova povprečja preko vseh podatkovnih množic, algoritmov za razvrščanje in zunanjih kazalcev.

5. Za vsak notranji kazalec in podatkovno množico izračunamo rang njene uspešnosti in nato povprečni rang preko vseh podatkovnih množic. Povprečni rang uporabimo za testiranje ničelne hipoteze, da so povprečni rangi enaki. Uporabimo Friedmanov neparametrični test [107]. Če statistični test zavrne ničelno hipotezo, potem s Shafferjevim testom ugotavljamo statistično pomembne razlike med pari metod.

Če na kratko povzamemo rezultate primerjave notranjih kazalcev, lahko rečemo naslednje. Pri stopnjevanju zahtevnosti podatkovnih množic `Complex2D` se opazno spreminja tudi vrstni red rangiranih kazalcev. Dunnov kazalec, njegova posplošena različica DNg in predlagani kazalec DNs so pridobili glede na ostale, ko smo povečali stopnjo linearne neločljivosti in zmanjšali medsebojno razdaljo med gručami. V primerih podatkov o izraženih genih v množicah `GENE` je opazen izrazit padec korelacije med primerjanimi notranjimi in referenčnimi zunanjimi kazalci. Razlog je najbrž v tem, da so genetski podatki zaradi svoje velike razsežnosti in majhnega števila vzorcev precej trši oreh kot dvorazsežni podatki. Kazalec DNs se izkaže kot eden izmed boljših v primerjavi na genetskih podatkih in je boljši od Dunnovega kazalca, čeprav ne izrazito. Podobno je v primeru ostalih realnih podatkov `REAL`, kjer je DNs statistično pomembno boljši od Dunnovega kazalca. Rezultati primerjave nam tudi postrežejo z izborom petih kazalcev, ki se v splošnem dobro obnesejo na omenjenih podatkih: kazalec CON [35], SEP in SEPmax [55], I [46] in predlagani DNs.

C.4 Samo-organizirajoča mreža povezana z gravitacijskim razvrščanjem

Samo-organizirajoča mreža (SOM) [119, 120] je razširjena metoda umetne nevronske mreže, ki se uči na nenadzorovan način, in se pogosto uporablja v povezavi z razvrščanjem podatkov [116]. V prvi vrsti je SOM učinkovito orodje za vizualizacijo podatkov, saj na nelinearen način preslika vhodne podatke v dvo- ali trirazsežni prostor, pri čemer ohranja topologijo podatkov. SOM uporablja algoritem tekmovalnega učenja, kar pomeni, da nevroni tekmujejo med seboj pri pokrivanju vhodnega prostora podatkov. Svoje uteži prilagajajo porazdelitvi vzorcev podatkov in tako se raztegnejo preko vhodnega prostora kot elastična mreža. Poleg tega je metoda SOM uporabna tudi pri razvrščanju, saj naučena mreža nevronov predstavlja prvo plast abstrakcije podatkov in lahko služi za ročno odkrivanje gruč v podatkih [117]. Ta postopek lahko avtomatiziramo z algoritmi za razvrščanje, ki nevrone združijo v gruče.

V disertaciji predlagamo nov pristop k združevanju nevronov mreže SOM, ki temelji na poenostavljeni simulaciji Newtonovega zakona o gravitaciji, ki pravi, da je privlačna sila med dvema telesoma sorazmerna s produktom njunih mas in obratno sorazmerna s kvadratom razdalje med njima. Naš namen je v čim večji meri izkoristiti prednosti obeh pristopov – tako SOM kot tudi gravitacijskega razvrščanja – da bi tako razvili učinkovit algoritem, ki bi bil sposoben najti gruče zapletenih oblik in samodejno določiti njihovo število. Algoritem za razvrščanje podatkov po principu gravitacije je prvi predlagal Wright leta 1977 [17]. Vsak podatkovni vzorec predstavlja masni delec, ki se pod vplivom privlačnih sil vseh drugih delcev premika po prostoru. Simulacija vključuje izračun pospeška in hitrosti vsakega delca posebej. Ko se dva delca dovolj približata drug drugemu, se združita v en delec. Leta 2003 so Gomez idr. predstavili poenostavitev Wrightovega algoritma, s ciljem zmanjšati njegovo časovno zahtevnost in izboljšati odpornost na šum [118]. Njihov algoritem RGC (ang. randomized gravitational clustering) uporablja poenostavljene enačbe za premikanje delcev, brez računanja hitrosti in pospeškov. Znatno pohitritev prinese tudi zmanjšanje števila korakov pri računanju rezultante sil na določen delec, saj se namesto vseh ostalih delcev upošteva samo en naključno izbran delec. V svoji raziskavi so Gomez idr. pokazali, da algoritem RGC deluje zadovoljivo, četudi upoštevajo samo 20 % vzorcev. Poleg tega algoritem samodejno določi število gruč. Zaradi teh lastnosti je algoritem RGC videti kot nalašč za združevanje nevronov mreže SOM in smo ga v prilagojeni obliki uporabili pri snovanju predlaganega algoritma za gravitacijsko razvrščanje samo-organizirajoče mreže (gSOM).

Vsak nevron mreže SOM, ki je s svojo utežjo najbližji vsaj enemu vzorcu podatkov, predstavlja masni delec. Ta je pod vplivom gravitacijskega polja drugih delcev, zato se premika po prostoru in se združuje z drugimi, če so dovolj blizu. Na ta način se tvorijo gruče in hkrati tudi hierarhija delcev znotraj njih, kar je še posebej uporabno v bioloških in socioloških raziskavah. Algoritem RGC je prilagojen tako, da upošteva sosednost delcev, ki izvira iz naučene mreže SOM: pri računanju privlačne sile na neki delec se z določeno verjetnostjo izbira med neposrednimi sosedi v mreži, sicer pa se izbira med vsemi preostalimi, ki niso sosedje. Na ta način upoštevamo topologijo podatkov in njihovo lokalnost, kar pripomore k odkrivanju gruč zapletenih oblik.

Algoritem gSOM smo zasnovali v dveh različicah: prva samodejno določa število gruč v podatkih, druga pa to število pričakuje od uporabnika. Obe različici smo primerjali s sedmimi drugimi algoritmi za razvrščanje na osnovi mreže SOM. Uporabili smo iste podatkovne množice kot drugje v disertaciji, tj. umetne, genetske in druge iz poskusov v realnem svetu. Primerjani algoritmi [2] so:

- SOM + metoda *k* voditeljev (SOMKm) [117],
- SOM + metoda *k* voditeljev s samodejnim določanjem števila gruč z Davies-Bouldinovim kazalcem veljavnosti (SOMKm*) [117],
- metoda Clusot* z rekurzivnim poplavljanjem [144],
- SOM + spektralni algoritem za razvrščanje (SOMSpec) [103, 156],
- SOM + spektralni algoritem za razvrščanje s samodejnim določanjem števila gruč (SOMSpec*) [103],
- SOM + iskanje povezanih komponent v mreži SOM (SOMStar*) [138],
- SOM + metoda normaliziranih rezov (SOMNcut) [160].

Sledili smo uveljavljeni metodologiji za primerjavo algoritmov strojnega učenja na več podatkovnih množicah in uporabili statistične teste za ugotavljanje pomembnih razlik med devetimi algoritmi v primerjavi [104–106]. Eden večjih izzivov v primerjavi je optimizacija parametrov, ki jih ima vsak algoritem. Po zgledu obširne primerjave algoritmov za razvrščanje na biomedicinskih podatkih [182] smo določili tri načine oziroma protokole za nastavitev parametrov:

- Protokol eCVI: uporablja zunanje kazalce veljavnosti razvrstitev, pri čemer predpostavljamo, da je ciljna pravilna razvrstitev podana. Za vsako podatkovno množico so izbrani tisti parametri algoritma, ki dajo najboljšo vrednost zunanjega kazalca. To pomeni, da rezultati tega protokola prikazujejo zgornjo mejo sposobnosti algoritma – torej, kakšen rezultat bi dosegel algoritem, če bi uganili idealne vrednosti njegovih parametrov. Uporabili smo tri zunanje kazalce: prilagojen Randov kazalec (ARI), prilagojen kazalec medsebojne informacije (AMI) in kazalec uravnotežene natančnosti razvrstitve (BCA).

[2]Algoritmi z zvezdico v imenu so zmožni sami odkriti število gruč v podatkih.

- Protokol CV: zasnovan je na prečnem preverjanju (ang. cross-validation) preko podatkovnih množic. Predpostavlja, da poznamo ciljno razvrstitev za podmnožico vseh podatkovnih množic v domeni. Ko nastavljamo parametre algoritma na neki množici podatkov, najprej po protokolu eCVI optimiziramo te parametre na vseh ostalih množicah podatkov in tiste vrednosti, ki se v povprečju najbolje obnesejo, uporabimo na obravnavani množici podatkov. Protokol CV torej predpostavlja, da je možno pridobljeno znanje iz ene množice podatkov, tj. vrednosti parametrov, prenesti na drugo. To pomeni, da si morajo biti množice podatkov znotraj neke domene dovolj podobne med seboj.

- Protokol CVI: za razliko od protokola eCVI, se tu uporabijo notranji kazalci veljavnosti, da z njimi ocenimo razvrstitve, ki jih algoritem izdela pri določeni vrednosti svojih parametrov. Notranji kazalec ne pozna ciljne razvrstitve, zato je ta protokol primeren za prikaz zmogljivosti algoritma v najpogostejši situaciji v resničnem svetu.

Glede na rezultate empirične primerjave algoritmov lahko povzamemo, da ni statistično pomembnih razlik med obema različicama algoritma gSOM, tj. med tisto, ki samodejno določi število gruč in tisto, ki to število prejme kot vhodni parameter. Vsekakor pa ima slednja v splošnem višjo povprečno oceno zmogljivosti, ne glede na izbran protokol optimizacije parametrov. Ugotavljamo, da je uspešnost algoritma gSOM zadovoljiva, posebno na podatkih o izražanju genov, in da med njim in najboljšimi ni statistično pomembnih razlik. Glavna prednost pred njegovim tesnim tekmecem, algoritmom SOMSpec, je občutno nižja časovna zahtevnost. S pomočjo protokola CV smo za vse algoritme v primerjavi določili najboljše nabore vrednosti parametrov za vsako problemsko domeno posebej in jih predlagamo kot privzete vrednosti.

C.5 Metoda uteženega ansambla

Razvrščanje z metodo ansambla je področje, ki se v zadnjih 15 letih intenzivno razvija. Eno izmed prvih študij na to temo je objavila Fred leta 2001 [198]. V njej je poizkušala z združevanjem več razvrstitev v sporazumno rešitev in predlagala glasovalni sistem, imenovan *kopičenje dokazov* (ang. evidence accumulation, EAC), kjer vsak par vzorcev podatkov dobi toliko glasov, kolikokrat je bil razvrščen v skupno gručo, gledano preko vseh razvrstitev v ansamblu. Metoda kopičenja dokazov je osnova velikega števila izpeljank in ena izmed dveh glavnih pristopov k združevanju razvrstitev v sporazum. Drugi pristop je osnovan na iskanju mediane razvrstitev [197], tj. razvrstitve, ki naj bi bila najboljši predstavnik ansambla. V tej disertaciji se posvečamo samo metodi kopičenja dokazov.

Postopek razvrščanja z ansambli je sestavljen iz dveh glavnih delov: gradnja ansambla z uporabo algoritmov za razvrščanje, ki jih v tem kontekstu imenujemo enostavni algoritmi; in uporaba združevalne funkcije, ki na podlagi razvrstitev v ansamblu izračuna sporazumno razvrstitev. Pokazano je bilo, da sta raznolikost in natančnost razvrstitev v ansamblu ključna dejavnika, ki vplivata na kakovost sporazumne razvrstitve [202, 205–207]. Raznolikost lahko zagotovimo na več načinov:

- z manipulacijo nad podatki, kar vključuje:
  - naključno vzorčenje glede na vzorce podatkov ali glede na njihove značilnice,
  - projekcijo podatkov na podprostore in
  - različno predstavitev podatkov;
- z manipulacijo algoritmov za razvrščanje. To zajema:
  - izbiro različnih algoritmov, tj. heterogeni tip ansambla, in
  - spreminjanje parametrov enega algoritma, tj. homogeni tip ansambla.

Odločili smo se za uporabo naključnega vzorčenja značilnic podatkov v kombinaciji s homogenim tipom ansambla. Ko je ansambel zgrajen, uporabimo združevalno funkcijo, da dobimo sporazumno razvrstitev. V splošnem lahko pričakujemo, da bo ta razvrstitev boljša od povprečja razvrstitev v ansamblu in bo odporna na šum, torej na slabe razvrstitve. Da bi združevalni funkciji olajšali delo, je bil v literaturi predlagan vmesni korak, ki uteži razvrstitve glede na njihovo kakovost [10]. Ta korak imenujemo analiza pomembnosti razvrstitev (ang. partition relevance analysis, PRA). Tako naj bi razvrstitve slabe kakovosti dobile majhne uteži, kar pomeni, da bodo manj prispevale k določanju sporazumne razvrstitve. Po našem vedenju so Duarte idr. leta 2005 prvi predlagali sistem uteževanja ansambla [13]. Gradili so na osnovi kopičenja dokazov in uporabili notranje kazalce veljavnosti za določanje uteži razvrstitev v ansamblu. To je tudi izhodišče za naše raziskave, kjer smo naslovili dve vprašanji: koliko in katere kazalce veljavnosti razvrstitev izbrati? Naš cilj je bil zgraditi sistem, ki bo to določal samodejno.

Jedro metode kopičenja dokazov je matrika podobnosti **S**, v kateri vsak element predstavlja podobnost med dvema vzorcema podatkov. Ta podobnost je izračunana kot število pojavitev teh dveh vzorcev v isti gruči. Če imamo na voljo še uteži posamezne razvrstitve v ansamblu, jih uporabimo za izračun utežene vsote pojavitev parov vzorcev v gručah. Matriko **S** nato uporabimo kot vhod v poljuben algoritem za razvrščanje, da dobimo želeno število gruč in tako sporazumno razvrstitev. Naš glavni prispevek na tem področju je razširitev postopka PRA z dodatnim korakom redukcije, kar imenujemo PRAr. Računanje uteži po postopku PRAr je tako sestavljeno iz štirih korakov:

1. Ocenjevanje razvrstitev v ansamblu z uporabo notranjih kazalcev veljavnosti. V poskusu smo jih uporabili 34.

2. Normalizacija in poenotenje vrednosti kazalcev. Definirali smo 4 funkcije za poenotenje.

3. Redukcija poenotenih vrednosti kazalcev. To pomeni zmanjševanje števila kazalcev s pomočjo algoritmov za izbiro in izločanje značilnic (ang. feature selection, extraction). Upamo, da s tem izločimo odvečne kazalce ali pa take, ki predstavljajo šum. Uporabili smo 3 metode za izbiro in 2 za izločanje značilnic. Omenjene metode zmanjšajo število značilnic na vrednost, ki jo določi algoritem DANCo za ocenitev notranje razsežnosti podatkov [237].

4. Izračun uteži razvrstitev. Definirali smo 5 funkcij.

V poskusih smo obravnavali vse možne kombinacije funkcij za poenotenje, redukcijo in izračun uteži in analizirali njihov vpliv na končni rezultat. Pri tem smo postopek PRAr uporabili na treh združevalnih funkcijah, imenovanih EAC, CSPA in DICLENS. Tako spremenjene funkcije smo preimenovali v EAC-W, CSPA-W in DICLENS-W. V primerjavi je sodelovalo naslednjih 12 združevalnih funkcij:

- kopičenje dokazov (EAC) [198],
- razvrščanje na osnovi podobnosti med gručami (CSPA) [192],
- razvrščanje na osnovi hipergrafa (HGPA) [192],

- razvrščanje z meta-gručami (MCLA) [192],
- povprečno uteženo kopičenje dokazov (JWEAC) [14],
- verjetnostno kopičenje dokazov (PAC) [221],
- uteženo kopičenje dokazov (WEA) [224],
- metoda ansamblov osnovana na povezavah (LCE) [211],
- cepitvena metoda ansamblov s samodejnim določanjem števila gruč (DICLENS) [220],
- EAC s postopkom PRAr (EAC-W),
- CSPA s postopkom PRAr (CSPA-W),
- DICLENS s postopkom PRAr (DICLENS-W).

Prispevke smo ovrednotili na več ravneh: primerjali smo enostavne algoritme za razvrščanje, med njimi tudi algoritem gSOM; primerjali smo združevalne funkcije; primerjali smo enostavne algoritme z metodami ansamblov. Primerjave so bile opravljene na istih podatkovnih množicah kot drugje v disertaciji. Za gradnjo ansamblov smo uporabili 6 enostavnih algoritmov: hierarhični algoritmi z minimalno (SL), maksimalno (CL) in povprečno metodo (AL) [50], metoda k voditeljev (KM) [102], spektralni algoritem za razvrščanje (Sp) [103] in algoritem gSOM. Raznolikost v ansamblu smo zagotovili s tremi načini vzorčenja značilnic – naključno smo izbrali 25-50 % ali 75-85 % značilnic oziroma smo ohranili vse – in s tremi načini izbire želenega števila gruč: pravo število gruč, določeno glede na ciljno razvrstitev; število gruč izbrano naključno med 2 in $\lceil \sqrt{N} \rceil$, kjer je N število vzorcev; in število gruč izbrano naključno na intervalu $[\lceil \sqrt{N} \rceil, \lceil \sqrt{N} \rceil + 20]$. Za vsako kombinacijo enostavnega algoritma za gradnjo ansambla, vzorčenja značilnic in izbire števila gruč smo zgradili ansambel 20 razvrstitev. Celoten postopek gradnje ansambla in združevanja v sporazum smo ponovili 30-krat in rezultate ocenili z zunanjimi kazalci veljavnosti ter izračunali povprečno oceno.

Iz rezultatov vidimo, da velika večina metod ansamblov dosega boljše povprečne rezultate kot enostavni algoritmi za razvrščanje. Opazili smo, da veliko vlogo igra način gradnje ansambla, natančneje, kateri enostavni algoritem za razvrščanje izberemo, koliko značilnic vzorčimo in koliko je želeno število gruč v sporazumni razvrstitvi. Vpliv nastavitve celotnega cevovoda razvrščanja z ansambli smo poglobljeno analizirali z uporabo Bayesovega statističnega modeliranja. Iz primerjave šestih enostavnih algoritmov za razvrščanje je razvidno, da se algoritem gSOM v večini primerov po zmogljivosti opazno ne razlikuje od najbolje uvrščenih. Najbolje se obnese na genetskih podatkih, kjer tudi izkazuje statistično pomembno boljšo zgornjo mejo zmogljivosti od vseh drugih. Dodatno prednost mu daje njegova majhna časovna zahtevnost, ki je linearna glede na število vzorcev v podatkovni množici. Nadalje smo ugotovili, da so razlike med različnimi združevalnimi funkcijami zelo majhne in zelo pogosto statistično neizrazite. Morda razlog tiči v dejstvu, da vse razen HGPA in MCLA temeljijo na principu kopičenja dokazov. Vključitev postopka PRAr se izkaže kot dobrodošlo, saj so metode ansamblov EAC-W, CSPA-W in DICLENS-W pogosto dosegale boljše povprečje ocen kot njihove različice brez vključenega koraka PRAr. Pri tem pa velja opomniti, da statistični testi niso našli izrazitih razlik med metodami s PRAr in tistimi brez.

C.6 Zaključek

V disertaciji smo se sprehodili skozi celoten proces razvrščanja podatkov v gruče s poudarkom na sodobnih pristopih, ki vključujejo metode ansamblov. Ločeno smo razpravljali o enostavnih algoritmih za razvrščanje in o vrednotenju razvrstitev – nato smo obe področji obravnavali kot sestavna dela ogrodja za razvrščanje podatkov z metodami uteženih ansamblov. V obširnem eksperimentalnem delu na podatkih iz več različnih domen smo primerjali metode razvrščanja z ugotavljanjem statistično pomembnih razlik med njimi.

Ocenjevanje razvrstitev smo predstavili kot ključno v celotnem cevovodu razvrščanja podatkov. Predlagali smo novo različico Dunnovega notranjega kazalca veljavnosti in ga poimenovali kazalec DNs. Zasnovan je na iskanju najkrajših poti v Gabrielovem grafu, ki med sabo povezuje vzorce podatkov. Vključuje tudi funkcijo kaznovanja velikega števila gruč, saj je to v praksi neželeno. Iz literature smo izbrali 33 že uveljavljenih kazalcev in jih primerjali s kazalcem DNs na umetnih in realnih podatkih. Sistematično smo ugotavljali obnašanje na podatkih z različnimi stopnjami linearne ločljivosti med gručami. Razpravljali smo tudi o tem, kako sploh meriti uspešnost kazalcev veljavnosti in opisali ter uporabili štiri dopolnjujoče metodologije. Naša študija je ena izmed najobširnejših v literaturi, če upoštevamo število primerjanih kazalcev, podatkovnih množic in mer uspešnosti. Iz rezultatov sklepamo, da je kazalec DNs večinoma boljši od svojega predhodnika, Dunnovega kazalca. Poleg tega se kazalec DNs konsistentno pojavlja med najboljšimi kazalci v primerjavi.

V nadaljevanju smo pregledali literaturo in poudarili ključna dela v kontekstu razvrščanja z uporabo dvostopenjskega postopka, ki temelji na samo-organizirajoči mreži (SOM). Razvili smo nov algoritem, v katerem sta močno sklopljena postopka samo-organizirajoče mreže in gravitacijskega razvrščanja. Predlagan algoritem gSOM samodejno odkrije število gruč v podatkih, ima nizko časovno zahtevnost glede na število vzorcev podatkov in poleg razvrstitve podatkov ponudi tudi informacijo o hierarhiji gruč, kar je uporabniku lahko v pomoč pri boljšem razumevanju strukture podatkov. V primerjavi z drugimi algoritmi za razvrščanje, tako konvencionalnimi kot tudi osnovanimi na samo-organizirajoči mreži, se algoritem gSOM izkaže kot primeren in obetaven, še zlasti na genetskih podatkih.

Metode razvrščanja z ansamblom so že dodobra uveljavljene in poznane po svoji boljši natančnosti in stabilnosti. V disertaciji smo dali poudarek na utežene pristope razvrščanja z ansambli, še posebno na različice združevalnih funkcij, ki temeljijo na principu kopičenja dokazov. Svoje raziskovanje smo gradili na obetavnih rezultatih študij, ki so predlagale uteževanje razvrstitev v ansamblu z uporabo kazalcev veljavnosti. Predlagali smo nadgradnjo tega pristopa, ki smo ga poimenovali analiza pomembnosti razvrstitev z redukcijo (PRAr). Empirično smo pokazali, da redukcija začetne velike množice kazalcev olajša celoten proces uteževanja, predvsem, če nam uspe odstraniti kazalce, ki nosijo malo informacije. Izdelali smo postopek redukcije, ki samodejno ugotovi potrebno število kazalcev, pri čemer uporabi cenilko notranje razsežnosti podatkov. Nato s pomočjo nenadzorovanih metod izbire in izločevanja značilnic izračuna uteži razvrstitev. Opisani sistem PRAr smo uporabili v povezavi s tremi združevalnimi funkcijami EAC, CSPA in DICLENS ter izvedli obširno primerjalno študijo, ki je vsebovala veliko različnih načinov gradnje ansamblov kot tudi različne nastavitve procesa uteževanja. Svoja opažanja lahko strnemo v naslednje točke:

- Metode ansamblov izboljšajo povprečno uspešnost enostavnih algoritmov za razvrščanje.

- Med dvanajstimi združevalnimi funkcijami v primerjavi smo našli zelo malo statistično pomembnih razlik. Tak rezultat je pričakovan, saj je večina funkcij osnovanih na istem principu – na kopičenju dokazov.

- Uporaba koraka redukcije v sistemu analize pomembnosti razvrstitev PRAr se je izkazala kot koristna. Metode EAC-W, CSPA-W in DICLENS-W, ki so delovale s sistemom PRAr so pogosto pokazale enako dobro ali boljšo zmogljivost kot različice brez PRAr, čeprav o statistično pomembnih razlikah ne moremo govoriti.

C.6.1 Nadaljnje delo

V vseh letih razvijanja in izboljševanja algoritmov za razvrščanje podatkov v gruče smo se naučili, da je to zgodba, ki se nikoli ne konča. Razvrščanje je slabo opredeljen problem, kar pomeni, da dopušča veliko možnih rešitev in interpretacij, ki so odvisne od tega, kako opredelimo pojem *dobre* razvrstitve [4]. Posledično je vedno dovolj dela na tem področju in veliko prostora za nadaljnje raziskave. Poglejmo si nekaj zanimivih smeri, ki bi jih bilo vredno obiskati v prihodnje.

Ena od pomanjkljivosti predstavljenega algoritma gSOM je njegova občutljivost na vrednost svojih parametrov. V tem oziru nas navdihuje nedavno objavljena študija Gomeza idr. [165], ki predlaga nekaj hevristik za samodejno nastavitev parametrov algoritma za gravitacijsko razvrščanje. Nadejamo se vključitve podobnih rešitev tudi v algoritem gSOM. Z rezultati slednjega smo zadovoljni, predvsem na podatkih o izražanju genov. Kakorkoli, obravnavali smo zgolj evklidsko razdaljo med vzorci in verjamemo, da bi bilo smotrno poskusiti še z drugimi merami podobnosti ali različnosti kot denimo s koeficientom korelacije ali s kosinusno razdaljo. Na to nas opozarja tudi raziskava o izbiri razdalj za razvrščanje genetskih podatkov, ki so jo opravili Jaskowiak idr. [244].

Naslednja ideja za nadaljnje delo je tudi povezana z genetskimi podatki. Ko smo ovrednotili zmogljivost 34 izbranih kazalcev veljavnosti razvrstitev na genetskih podatkih, smo pri vseh kazalcih opazili precej slabe rezultate. Domnevamo, da razlog tiči v dejstvu, da so uporabljeni kazalci *splošno-namenski* in niso posebej prilagojeni domeni genetike, kot so denimo kazalci, ki sta jih razvila Datta in Datta [48, 245]. Ti kazalci izkoriščajo dodatne informacije o funkcijskih razredih, kamor se določeni geni uvrščajo. Zanima nas, kakšni bi bili rezultati primerjave med temi kazalci in splošno-namenskimi in ali bi se slednje dalo izboljšati z uporabo omenjenih funkcijskih razredov.

V povezavi z metodami ansamblov imamo slutnjo, da je pristop kopičenja dokazov blizu svojih zgornjih meja zmogljivosti. Ta hipoteza ima podlago v rezultatih primerjave, kjer nismo našli skoraj nobenih statistično pomembnih razlik med ducatom algoritmov, temelječih na omenjenem pristopu. Kljub temu smo opazili izboljšanje rezultatov pri uporabi uteženih ansamblov s sistemom PRAr in mislimo, da bi bilo smiselno prenesti ta postopek tudi na metode ansamblov, ki ne temeljijo na kopičenju dokazov, temveč na iskanju mediane med razvrstitvami [10, 197].

BIBLIOGRAPHY

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Ian H. Witten, Eibe Frank, and Mark A. Hall. What's It All About? In *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 1, pages 3–38. Elsevier, 2011. doi: 10.1016/B978-0-12-374856-0.00001-8.

[3] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003. URL `http://lyle.smu.edu/{~}mhd/book`.

[4] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. doi: 10.1016/j.patrec.2009.09.011.

[5] Rui Xu and Donald Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141.

[6] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1990. doi: 10.1002/9780470316801.

[7] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. doi: 10.1145/331499.331504.

[8] Jon Kleinberg. An Impossibility Theorem for Clustering. In S Becker, S Thrun, and K Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 446–453. MIT Press, 2003.

[9] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In F Roli and J Kittler, editors, *Multiple Classifier Systems*, volume 1857, pages 1–15. Springer, 2000. doi: 10.1007/3-540-45014-9_1.

[10] Sandro Vega-Pons, Jyrko Correa-Morris, and José Ruiz-Shulcloper. Weighted partition consensus via kernels. *Pattern Recognition*, 43:2712–2724, 2010. doi: 10.1016/j.patcog.2010.03.001.

[11] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013. doi: 10.1016/j.patcog.2012.07.021.

[12] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of Internal Clustering Validation Measures. In *IEEE International Conference on Data Mining*, pages 911–916. IEEE, 2010. doi: 10.1109/ICDM.2010.35.

[13] F. Jorge Duarte, Ana L.N. Fred, André Lourenço, and M. Fátima Rodrigues. Weighting Cluster Ensembles in Evidence Accumulation Clustering. In *Portuguese Conference on Artificial Intelligence*, pages 159–167. IEEE, 2005. doi: 10.1109/EPIA.2005.341287.

[14] F. Jorge F. Duarte, Ana L.N. Fred, André Lourenço, and M. Fátima C. Rodrigues. Weighted Evidence Accumulation Clustering. In Simeon J. Simoff, Graham J. Williams, John Galloway, and Inna Kolyshkina, editors, *Fourth Australasian Conference on Knowledge Discovery and Data Mining*, pages 205—-220, 2005.

[15] Nejc Ilc and Andrej Dobnikar. Gravitational Clustering of the Self-Organizing Map. In Andrej Dobnikar, Uroš Lotrič, and Branko Šter, editors, *Adaptive and Natural Computing Algorithms*, volume 6594 of *Lecture Notes in Computer Science*, pages 11–20. Springer, 2011. doi: 10.1007/978-3-642-20267-4_2.

[16] Teuvo Kohonen. *Self-Organizing Maps*. Springer Berlin Heidelberg, 3rd edition, 2001. doi: 10.1007/978-3-642-56927-2.

[17] W.E. Wright. Gravitational clustering. *Pattern Recognition*, 9(3):151–166, 1977. doi: 10.1016/0031-3203(77)90013-9.

[18] Nejc Ilc and Andrej Dobnikar. Generation of a clustering ensemble based on a gravitational self-organising map. *Neurocomputing*, 96:47–56, 2012. doi: 10.1016/j.neucom.2011.10.043.

[19] J.C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1973. doi: 10.1080/01969727308546046.

[20] Nejc Ilc. Modified Dunn's cluster validity index based on graph theory. *Przeglad Elektrotechniczny (Electrical Review)*, 88(2):126–131, 2012. URL http://red.pe.org.pl/articles/2012/2/36.pdf.

[21] Alfred Ultsch. Clustering with SOM: U*C. In *Workshop on Self-Organizing Maps (WSOM 2005)*, pages 75–82, Paris, France, 2005.

[22] Yaling Pei and Osmar Zaïane. A synthetic data generator for clustering and outlier analysis. Technical report, Department of Computing science, University of Alberta, 2006.

[23] Joseph E. Hoag and Craig W. Thompson. A parallel general-purpose synthetic data generator. *ACM SIGMOD Record*, 36(1):19–24, 2007. doi: 10.1145/1276301.1276305.

[24] Alexander Alexandrov, Kostas Tzoumas, and Volker Markl. Myriad: scalable and expressive data generation. *Proceedings of the VLDB Endowment*, 5: 1890–1893, 2012. doi: 10.14778/2367502.2367530.

[25] Josh Eno and Craig W. Thompson. Generating synthetic data to match data mining patterns. *IEEE Internet Computing*, 12:78–82, 2008. doi: 10.1109/MIC.2008.55.

[26] Marko Robnik-Šikonja. Data Generators for Learning Systems Based on RBF Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 27(5):926–938, 2016. doi: 10.1109/TNNLS.2015.2429711.

[27] Iris Adä and Michael R. Berthold. The New Iris Data: Modular Data Generators. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 413–422, 2010. doi: 10.1145/1835804.1835858.

[28] Janick V. Frasch, Aleksander Lodwich, Faisal Shafait, and Thomas M. Breuel. A Bayes-true data generator for evaluation of supervised and unsupervised learning methods. *Pattern Recognition Letters*, 32(11): 1523–1531, 2011. doi: 10.1016/j.patrec.2011.04.010.

[29] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2000.

[30] David Elizondo. The linear separability problem: Some testing methods. *IEEE Transactions on Neural Networks*, 17(2):330–344, 2006. doi: 10.1109/TNN.2005.860871.

[31] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983. doi: 10.1109/TIT.1983.1056714.

[32] Herbert Edelsbrunner. Alpha Shapes – a Survey. *Tessellations in the Sciences*, pages 1–25, 2010.

[33] Herbert Edelsbrunner. Weighted alpha shapes. Technical report, University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

[34] Abbas Bahrololoum, Hossein Nezamabadi-pour, and Saeid Saryazdi. A data clustering approach based on universal gravity rule. *Engineering Applications of Artificial Intelligence*, 45:415–428, 2015. doi: 10.1016/j.engappai.2015.07.018.

[35] Julia Handl, Joshua Knowles, and Douglas B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

[36] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[37] Ibai Gurrutxaga, Javier Muguerza, Olatz Arbelaitz, Jesús M. Pérez, and José I. Martín. Towards a standard methodology to evaluate internal cluster validity indices. *Pattern Recognition Letters*, 32(3):505–515, 2011. doi: 10.1016/j.patrec.2010.11.006.

[38] Lucas Vendramin, Ricardo J.G.B. Campello, and Eduardo R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3:209–235, 2010. doi: 10.1002/sam.10080.

[39] Nikhil R. Pal and J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6): 847–857, 1997.

[40] James C. Bezdek and Nikhil R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics*, 28:301–315, 1998.

[41] Ariel E. Bayá and Pablo M. Granitto. How many clusters: a validation index for arbitrary-shaped clusters. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(2):401–14, 2013. doi: 10.1109/TCBB.2013.32.

[42] Julia Handl and Joshua Knowles. Exploiting the trade-off –the benefits of multiple objectives in data clustering. In Carlos Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 547–560. Springer Berlin / Heidelberg, 2005.

[43] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969. ISSN 00397989.

[44] David W. Matula and Robert R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis*, 12(3):205–222, 1980.

[45] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2001.

[46] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance Evaluation of Some Clustering Algorithms and Validity Indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1650–1654, 2002.

[47] S. Saitta, B. Raphael, and I.F.C. Smith. A comprehensive validity index for clustering. *Intelligent Data Analysis*, 12(6):529–548, 2008.

[48] Susmita Datta and Somnath Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4): 459–466, 2003. doi: 10.1093/bioinformatics/btg025.

[49] Pablo A. Jaskowiak, Davoud Moulavi, Antonio C.S. Furtado, Ricardo J.G.B. Campello, Arthur Zimek, and Jörg Sander. On strategies for building effective ensembles of relative clustering validity criteria. *Knowledge and Information Systems*, 2015. ISSN 0219-1377. doi: 10.1007/s10115-015-0851-6.

[50] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. doi: 10.1002/widm.53.

[51] T. Calinski and J. Harabasz. A Dendrite Method for Cluster Analysis. *Communications in Statistics*, 3(1): 1–27, 1974.

[52] Lawrence J. Hubert and Joel R. Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, 83(6):1072–1080, 1976.

[53] D.L. Davies and D.W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[54] Minho Kim and R.S. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353–2363, 2005.

[55] Roded Sharan. *Graph modification problems and their applications to genomic research*. PhD thesis, Tel-Aviv University, 2002.

[56] Malay K. Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487–501, 2004.

[57] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: finding the optimal partitioning of a data set. In *Proceedings IEEE International Conference on Data Mining*, pages 187–194. IEEE, 2001.

[58] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[59] Sara Dolničar, Klaus Grabler, and Josef A. Mazanec. A tale of three cities: perceptual charting for analysing destination images. In A.G. Woodside, G.I. Crouch, J.A. Mazanec, M. Oppermann, and M.Y. Sakai, editors, *Consumer psychology of tourism, hospitality and leisure*, pages 39–62. CABI Publishing, 1999.

[60] X.L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[61] Lars Dyrskjøt, Thomas Thykjaer, Mogens Kruhøffer, Jens Ledet Jensen, Niels Marcussen, Stephen Hamilton-Dutoit, Hans Wolf, and Torben F. Ørntoft. Identifying distinct classes of bladder carcinoma using microarrays. *Nature Genetics*, 33(1): 90–96, 2003. doi: 10.1038/ng1061.

[62] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J.A. Olson, J.R. Marks, and J.R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98(20):11462–11467, 2001. doi: 10.1073/pnas.201162998.

[63] Dondapati Chowdary, Jessica Lathrop, Joanne Skelton, Kathleen Curtin, Thomas Briggs, Yi Zhang, Jack Yu, Yixin Wang, and Abhijit Mazumder. Prognostic Gene Expression Signatures Can Be Measured in Tissues Collected in RNAlater Preservative. *The Journal of Molecular Diagnostics*, 8(1):31–39, 2006. doi: 10.2353/jmoldx.2006.050056.

[64] A.I. Su, J.B. Welsh, L.M. Sapinoso, S.G. Kern, P. Dimitrov, H. Lapp, P.G. Schultz, S.M. Powell, C.A. Moskaluk, H.F. Frierson, and G.M. Hampton. Molecular classification of human carcinomas by use of gene expression signatures. *Cancer Research*, 61(20): 7388–7393, 2001.

[65] Scott L. Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M. Sturla, Michael Angelo, Margaret E. McLaughlin, John Y. H. Kim, Liliana C. Goumnerova, Peter M. Black, Ching Lau, Jeffrey C. Allen, David Zagzag, James M. Olson, Tom Curran, Cynthia Wetmore, Jaclyn A. Biegel, Tomaso Poggio, Shayan Mukherjee, Ryan Rifkin, Andrea Califano,

Gustavo Stolovitzky, David N. Louis, Jill P. Mesirov, Eric S. Lander, and Todd R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870): 436–442, 2002. doi: 10.1038/415436a.

[66] John I. Risinger, G. Larry Maxwell, G.V.R. Chandramouli, Amir Jazaeri, Olga Aprelikova, Tricia Patterson, Andrew Berchuck, and J. Carl Barrett. Microarray analysis reveals distinct gene expression profiles among different histologic types of endometrial cancer. *Cancer Research*, 63(1):6–11, 2003.

[67] Yu Liang, Maximilian Diehn, Nathan Watson, Andrew W. Bollen, Ken D. Aldape, M. Kelly Nicholas, Kathleen R. Lamborn, Mitchel S. Berger, David Botstein, Patrick O. Brown, and Mark A. Israel. Gene expression profiling reveals molecularly and clinically distinct subtypes of glioblastoma multiforme. *Proceedings of the National Academy of Sciences*, 102(16): 5814–5819, 2005. doi: 10.1073/pnas.0402870102.

[68] Markus Bredel, Claudia Bredel, Dejan Juric, Griffith R. Harsh, Hannes Vogel, Lawrence D. Recht, and Branimir I. Sikic. Functional Network Analysis Reveals Extended Gliomagenesis Pathway Maps and Three Novel MYC-Interacting Genes in Human Gliomas. *Cancer Research*, 65(19):8679–8689, 2005. doi: 10.1158/0008-5472.CAN-05-1204.

[69] Catherine L. Nutt, D.R. Mani, Rebecca A. Betensky, Pablo Tamayo, J. Gregory Cairncross, Christine Ladd, Ute Pohl, Christian Hartmann, Margaret E. McLaughlin, Tracy T. Batchelor, Peter M. Black, Andreas von Deimling, Scott L. Pomeroy, Todd R. Golub, and David N. Louis. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Research*, 63(7):1602–1607, 2003.

[70] Xin Chen, Siu Tim Cheung, Samuel So, Sheung Tat Fan, Christopher Barry, John Higgins, Kin-Man Lai, Jiafu Ji, Sandrine Dudoit, Irene O.L. Ng, Matt Van De Rijn, David Botstein, and Patrick O. Brown. Gene expression patterns in human liver cancers. *Molecular biology of the cell*, 13(6):1929–39, 2002. doi: 10.1091/mbc.02-02-0023.

[71] Eng-Juh Yeoh, Mary E. Ross, Sheila A. Shurtleff, W. Kent Williams, Divyen Patel, Rami Mahfouz, Fred G. Behm, Susana C. Raimondi, Mary V. Relling, Anami Patel, Cheng Cheng, Dario Campana, Dawn Wilkins, Xiaodong Zhou, Jinyan Li, Huiqing Liu, Ching-Hon Pui, William E. Evans, Clayton Naeve, Limsoon Wong, and James R. Downing. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1(2): 133–143, 2002.

[72] Scott A. Armstrong, Jane E. Staunton, Lewis B. Silverman, Rob Pieters, Monique L. den Boer, Mark D. Minden, Stephen E. Sallan, Eric S. Lander, Todd R. Golub, and Stanley J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30(1): 41–47, 2002. doi: 10.1038/ng765.

[73] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999. doi: 10.1126/science.286.5439.531.

[74] A. Bhattacharjee, W.G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E.J. Mark, E.S. Lander, W. Wong, B.E. Johnson, T.R. Golub, D.J. Sugarbaker, and M. Meyerson. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences*, 98(24): 13790–13795, 2001. doi: 10.1073/pnas.191502998.

[75] M.E. Garber, O.G. Troyanskaya, K. Schluens, S. Petersen, Z. Thaesler, M. Pacyna-Gengelbach, M. van de Rijn, G.D. Rosen, C.M. Perou, R.I. Whyte, R.B. Altman, P.O. Brown, D. Botstein, and I. Petersen. Diversity of gene expression in adenocarcinoma of the lung. *Proceedings of the National Academy of Sciences*, 98(24):13784–13789, 2001. doi: 10.1073/pnas.241500798.

[76] Ash A. Alizadeh, Michael B. Eisen, R. Eric Davis, Chi Ma, Izidore S. Lossos, Andreas Rosenwald, Jennifer C. Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, John I. Powell, Liming Yang, Gerald E. Marti, Troy Moore, James Hudson, Lisheng Lu, David B. Lewis, Robert Tibshirani, Gavin Sherlock, Wing C. Chan, Timothy C. Greiner, Dennis D. Weisenburger, James O. Armitage, Roger Warnke, Ronald Levy, Wyndham Wilson, Michael R. Grever, John C. Byrd, David Botstein, Patrick O. Brown, and Louis M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000. doi: 10.1038/35000501.

[77] Margaret A. Shipp, Ken N. Ross, Pablo Tamayo, Andrew P. Weng, Jeffery L. Kutok, Ricardo C.T. Aguiar, Michelle Gaasenbeek, Michael Angelo, Michael Reich, Geraldine S. Pinkus, Tane S. Ray, Margaret A. Koval, Kim W. Last, Andrew Norton, T. Andrew Lister, Jill Mesirov, Donna S. Neuberg, Eric S. Lander, Jon C. Aster, and Todd R. Golub. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, 8(1):68–74, 2002. doi: 10.1038/nm0102-68.

[78] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, V. Sondak, N. Hayward, and J. Trent. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406(6795):536–540, 2000. doi: 10.1038/35020115.

[79] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, Sridhar Ramaswamy, William G. Richards, David J. Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer research*, 62(17):4963–7, 2002.

[80] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M. Loda, E.S. Lander, and T.R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences*, 98(26): 15149–15154, 2001. doi: 10.1073/pnas.211566398.

[81] Scott A. Tomlins, Rohit Mehra, Daniel R. Rhodes, Xuhong Cao, Lei Wang, Saravana M. Dhanasekaran, Shanker Kalyana-Sundaram, John T. Wei, Mark A. Rubin, Kenneth J. Pienta, Rajal B. Shah, and Arul M. Chinnaiyan. Integrative molecular concept modeling of prostate cancer progression. *Nature Genetics*, 39(1): 41–51, 2007. doi: 10.1038/ng1935.

[82] Jacques Lapointe, Chunde Li, John P. Higgins, Matt van de Rijn, Eric Bair, Kelli Montgomery, Michelle Ferrari, Lars Egevad, Walter Rayford, Ulf Bergerheim, Peter Ekman, Angelo M. DeMarzo, Robert Tibshirani, David Botstein, Patrick O. Brown, James D. Brooks, and Jonathan R. Pollack. Gene expression profiling identifies clinically relevant subtypes of prostate cancer. *Proceedings of the National Academy of Sciences*, 101(3):811–816, 2004. doi: 10.1073/pnas.0304146101.

[83] Dinesh Singh, Phillip .G Febbo, Kenneth Ross, Donald G. Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A. Renshaw, Anthony V. D'Amico, Jerome P. Richie, Eric S. Lander, Massimo Loda, Philip W. Kantoff, Todd R. Golub, and William R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209, 2002. doi: 10.1016/S1535-6108(02)00030-2.

[84] Javed Khan, Jun S. Wei, Markus Ringnér, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, Carsten Peterson, and Paul S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001. doi: 10.1038/89044.

[85] P. Laiho, A. Kokko, S. Vanharanta, R. Salovaara, H. Sammalkorpi, H. Järvinen, J.P. Mecklin, T.J. Karttunen, K. Tuppurainen, V. Davalos, S. Schwartz, D. Arango, M.J. Mäkinen, and L.A. Aaltonen. Serrated carcinomas form a subclass of colorectal cancer with distinct molecular basis. *Oncogene*, 26(2): 312–320, 2007. doi: 10.1038/sj.onc.1209778.

[86] Marcilio C.P. de Souto, Ivan G. Costa, Daniel S.A. de Araujo, Teresa B. Ludermir, and Alexander Schliep. Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics*, 9(1):497, 2008. doi: 10.1186/1471-2105-9-497.

[87] William H. Wolberg and Olvi L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87(23):9193–9196, 1990. doi: 10.1073/pnas.87.23.9193.

[88] L.I. Kuncheva, S.T. Hadjitodorov, and L.P. Todorova. Experimental Comparison of Cluster Ensemble Methods. In *9th International Conference on Information Fusion*, pages 1–7. IEEE, 2006. doi: 10.1109/ICIF.2006.301614.

[89] David Gil, Jose Luis Girela, Joaquin De Juan, M. Jose Gomez-Torres, and Magnus Johnsson. Predicting seminal quality with artificial intelligence methods. *Expert Systems with Applications*, 39(16):12564–12573, 2012. doi: 10.1016/j.eswa.2012.05.028.

[90] I.W. Evett and E.J. Spiehler. Rule induction in forensic science. Technical report, Central Research Establishment, Home Office Forensic Science Service, 1987.

[91] Vincent G. Sigillito, Simon P. Wing, Larrie V. Hutton, and Kile B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

[92] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

[93] Peter W. Frey and David J. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, 1991. doi: 10.1007/BF00114162.

[94] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[95] S. Aeberhard, D. Coomans, and O. de Vel. Comparison of Classifiers in High Dimensional Settings. Technical report, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, 1992.

[96] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009. doi: 10.1016/j.dss.2009.05.016.

[97] Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *International Conference on Intelligent Systems for Molecular Biology*, volume 4, pages 109–115, 1996.

[98] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985. doi: 10.1007/BF02294245.

[99] Shihong Yue, Jianpei Wang, Jeenshing Wang, and Xiujuan Bao. A new validity index for evaluating the clustering results by partitional clustering algorithms. *Soft Computing*, 20(3):1127–1138, 2015. doi: 10.1007/s00500-014-1577-1.

[100] Lucas Vendramin, Ricardo J.G.B. Campello, and Eduardo R. Hruschka. On the Comparison of Relative Clustering Validity Criteria. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 733–744. Society for Industrial and Applied Mathematics, 2009. doi: 10.1137/1.9781611972795.63.

[101] M.G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1):81–93, 1938. doi: 10.1093/biomet/30.1-2.81.

[102] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bulletin de L'Académie Polonaise des sciences*, 4 (12):801–804, 1956.

[103] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in neural information processing systems*, 2:1601–1608, 2004.

[104] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. URL `http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf`.

[105] Salvador García and Francisco Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[106] Aydın Ulaş, Olcay Taner Yıldız, and Ethem Alpaydın. Cost-conscious comparison of supervised learning algorithms over multiple data sets. *Pattern Recognition*, 45(4):1772–1781, 2012. doi: 10.1016/j.patcog.2011.10.005.

[107] Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[108] Juliet Popper Shaffer. Modified Sequentially Rejective Multiple Test Procedures. *Journal of the American Statistical Association*, 81(395):826, 1986. doi: 10.2307/2289016.

[109] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075.

[110] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[111] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi: 10.2307/2284239.

[112] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML'09*, pages 1073–1080. ACM Press, 2009. doi: 10.1145/1553374.1553511.

[113] Henry Carrillo, Kay H. Brodersen, and José A. Castellanos. Probabilistic performance evaluation for multiclass classification using the posterior balanced accuracy. In Manuel A Armada, Alberto Sanfeliu, and Manuel Ferre, editors, *ROBOT2013: First Iberian Robotics Conference SE - 25*, volume 252 of *Advances in Intelligent Systems and Computing*, pages 347–361. Springer International Publishing, 2014. doi: 10.1007/978-3-319-03413-3_25.

[114] Marina Meilă. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis*, 98:873–895, 2007.

[115] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2 (1-2):83–97, 1955. doi: 10.1002/nav.3800020109.

[116] K.L. Du. Clustering: A neural network approach. *Neural Networks*, 23(1):89–107, 2010. doi: 10.1016/j.neunet.2009.08.007.

[117] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000. doi: 10.1109/72.846731.

[118] Jonatan Gomez, Dipankar Dasgupta, and Olfa Nasraoui. A New Gravitational Clustering Algorithm. In Daniel Barbara and Chandrika Kamath, editors, *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 83–94. Society for Industrial and Applied Mathematics, 2003. doi: 10.1137/1.9781611972733.8.

[119] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325.

[120] Teuvo Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52–65, 2013. doi: 10.1016/j.neunet.2012.09.018.

[121] Teuvo Kohonen. Things you haven't heard about the self-organizing map. In *IEEE International Conference on Neural Networks*, pages 1147–1156. IEEE, 1993. doi: 10.1109/ICNN.1993.298719.

[122] Ning Chen, Bernardete Ribeiro, Armando Vieira, and An Chen. Clustering and visualization of bankruptcy trajectory using self-organizing map. *Expert Systems with Applications*, 40(1):385–393, 2013. doi: 10.1016/j.eswa.2012.07.047.

[123] Xi Chen, Zibin Zheng, Xudong Liu, Zicheng Huang, and Hailong Sun. Personalized QoS-Aware Web Service Recommendation and Visualization. *IEEE Transactions on Services Computing*, 6(1):35–47, 2013. doi: 10.1109/TSC.2011.35.

[124] Amnon Frenkel, Edward Bendit, and Sigal Kaplan. The linkage between the lifestyle of knowledge-workers and their intra-metropolitan residential choice: A clustering approach based on self-organizing maps. *Computers, Environment and Urban Systems*, 39:151–161, 2013. doi: 10.1016/j.compenvurbsys.2012.09.001.

[125] Fengxian He, W.X. Shen, Qiang Song, Ajay Kapoor, Demon Honnery, and Daya Dayawansa. Clustering LiFePO4 cells for battery pack based on neural network in EVs. In *IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, pages 1–5. IEEE, 2014. doi: 10.1109/ITEC-AP.2014.6941056. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6941056`.

[126] Wen Bo Wang, Mao Lin Huang, Jinson Zhang, and Wei Lai. Detecting Criminal Relationships through SOM Visual Analytics. In *2015 19th International Conference on Information Visualisation*, pages 316–321. IEEE, 2015. doi: 10.1109/iV.2015.62.

[127] Pei Zhou, Jinliang Huang, Robert Gilmore Pontius, and Huasheng Hong. New insight into the correlations between land use and water quality in a coastal watershed of China: Does point source pollution weaken it? *Science of The Total Environment*, 543:591–600, 2016. doi: 10.1016/j.scitotenv.2015.11.063.

[128] Yonggang Liu, Robert H. Weisberg, Stefano Vignudelli, and Gary T. Mitchum. Patterns of the Loop Current System and Regions of Sea Surface Height Variability in the Eastern Gulf of Mexico Revealed by the Self-Organizing Maps. *Journal of Geophysical Research: Oceans*, 2016. doi: 10.1002/2015JC011493.

[129] I.L. Hudson, S.Y. Leemaqz, A.T. Neffe, and A.D. Abell. Classifying Calpain Inhibitors for the Treatment of Cataracts: A Self Organising Map (SOM) ANN/KM Approach in Drug Discovery. In Subana Shanmuganathan and Sandhya Samarasinghe, editors, *Artificial Neural Network Modelling*, pages 161–212. Springer International Publishing, 2016. doi: 10.1007/978-3-319-28495-8_9.

[130] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Self-Organizing Map in Matlab: the SOM Toolbox. In *In Proceedings of the Matlab DSP Conference*, pages 35–40, 1999. URL `http://www.cis.hut.fi/somtoolbox/`.

[131] P.V. Balakrishnan, Martha C. Cooper, Varghese S. Jacob, and Phillip A. Lewis. A study of the classification capabilities of neural networks using unsupervised learning: A comparison with K-means clustering. *Psychometrika*, 59(4):509–525, 1994. doi: 10.1007/BF02294390.

[132] Shaw K. Chen, Paul Mangiameli, and David West. The comparative ability of self-organizing neural networks to define cluster structure. *Omega*, 23(3): 271–279, 1995. doi: 10.1016/0305-0483(95)00011-C.

[133] Jianchang Mao and Anil K. Jain. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, 7(1):16–29, 1996. doi: 10.1109/72.478389.

[134] Sueli A. Mingoti and Joab O. Lima. Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research*, 174(3): 1742–1759, 2006. doi: 10.1016/j.ejor.2005.03.039.

[135] Nikhil R. Pal, James C. Bezdek, and Eric C.K. Tsao. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):549–557, 1993. doi: 10.1109/72.238310.

[136] Melody Y. Kiang. Extending the Kohonen self-organizing map networks for clustering analysis. *Computational Statistics & Data Analysis*, 38(2):161–180, 2001. doi: 10.1016/S0167-9473(01)00040-8.

[137] Alfred Ultsch and Peter H. Simeon. Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. In Bernard Widrow and Bernard Angeniol, editors, *Proceedings of the International Neural Network Conference*, pages 305–308, 1990.

[138] Lutz Hamel and C.W. Brown. Improved Interpretability of the Unified Distance Matrix with Connected Components. In *7th International Conference on Data Mining*, pages 338–343, 2011.

[139] Elena V. Samsonova, Joost N. Kok, and Ad P. Ijzerman. TreeSOM: Cluster analysis in the self-organizing map. *Neural networks*, 19(6-7):935–49, 2006. doi: 10.1016/j.neunet.2006.05.003.

[140] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991. doi: 10.1109/34.87344.

[141] José Alfredo Ferreira Costa and Márcio L. de Andrade Netto. Cluster analysis using self-organizing maps and image processing techniques. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 367–372, 1999.

[142] José Alfredo Ferreira Costa and Márcio L. de Andrade Netto. Clustering of complex shaped data sets via Kohonen maps and mathematical morphology. In *Proceedings of the SPIE, Data Mining and Knowledge Discovery*, pages 16–27, 2001. doi: 10.1117/12.421088.

[143] Martin Bogdan and Wolfgang Rosenstiel. Detection of Custer in Self-Organizing Maps for Controlling a Prostheses using Nerve Signals. In *European Symposium on Artificial Neural Networks*, pages 131–136, 2001.

[144] Dominik Brugger, Martin Bogdan, and Wolfgang Rosenstiel. Automatic Cluster Detection in Kohonen's SOM. *IEEE Transactions on Neural Networks*, 19(3):442–459, 2008. doi: 10.1109/TNN.2007.909556.

[145] David Opolon and Fabien Moutarde. Fast semi-automatic segmentation algorithm for Self-Organizing Maps. In *ESANN'2004*, pages 507–512, 2004.

[146] Fabien Moutarde and Alfred Ultsch. U*F clustering: a new performant cluster-mining method based on segmentation of Self-Organizing Maps. In *Workshop on SOM*, pages 25–32, 2005.

[147] Alfred Ultsch and F. Mörchen. ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. Technical report, Dept. of Mathematics and Computer Science, University of Marburg, 2005.

[148] Alfred Ultsch. Emergence in Self Organzing Feature Maps. In *Proceedings of the 6th International Workshop on Self-Organizing Maps 2007*, pages 1–7, 2007.

[149] Aaron M. Newman and James B. Cooper. AutoSOME: a clustering method for identifying gene expression modules without prior knowledge of cluster number. *BMC Bioinformatics*, 11:117, 2010. doi: 10.1186/1471-2105-11-117.

[150] José Alfredo Ferreira Costa and Hujun Yin. Gradient-based SOM clustering and visualisation methods. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010. doi: 10.1109/IJCNN.2010.5596623.

[151] José Alfredo Ferreira Costa. Clustering and visualizing SOM results. In *Intelligent Data Engineering and Automated Learning - IDEAL 2010*, pages 334–343, 2010.

[152] Jouko Lampinen and Erkki Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(2-3):261–272, 1992. doi: 10.1007/BF00118594.

[153] F. Murtagh. Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering. *Pattern Recognition Letters*, 16(4):399–408, 1995. doi: 10.1016/0167-8655(94)00113-H.

[154] Cheng-ching Chang and Ssu-han Chen. A comparative analysis on artificial neural network-based two-stage clustering. *Cogent Engineering*, 2(1): 995785, 2015. doi: 10.1080/23311916.2014.995785.

[155] Sitao Wu and Tommy W.S. Chow. Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density. *Pattern Recognition*, 37(2):175–188, 2004. doi: 10.1016/S0031-3203(03)00237-1.

[156] Kadim Taşdemir. Spectral Clustering as an Automated SOM Segmentation Tool. In J. Laaksonen and T. Honkela, editors, *Advances in Self-Organizing Maps*, pages 71–78. Springer-Verlag Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-21566-7_7.

[157] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14: 849–856, 2002.

[158] Kadim Taşdemir. Vector quantization based approximate spectral clustering of large datasets. *Pattern Recognition*, 45(8):3034–3044, 2012. doi: 10.1016/j.patcog.2012.02.012.

[159] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[160] Le Yang, Zhongbin Ouyang, and Yong Shi. A Modified Clustering Method Based on Self-Organizing Maps and Its Applications. *Procedia Computer Science*, 9:1371–1379, 2012. doi: 10.1016/j.procs.2012.04.151.

[161] Zhiwen Yu, Jane You, Guoqiang Han, Le Li, and Xiaowei Wang. Fast normalized cut algorithm based on self-organizing map. In *International Conference on Machine Learning and Cybernetics*, pages 15–17, 2012.

[162] Leonardo Enzo Brito da Silva and José Alfredo Ferreira Costa. A Density-Based Clustering of the Self-Organizing Map Using Graph Cut. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 32–40, 2014. doi: 10.1109/CIDM.2014.7008145.

[163] Leonardo Enzo Brito da Silva and José Alfredo Ferreira Costa. Clustering of the self-organizing map using particle swarm optimization and validity indices. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3798–3806, 2014. doi: 10.1109/IJCNN.2014.6889954.

[164] James S. Coleman. Clustering in N dimensions by use of a system of forces. *The Journal of Mathematical Sociology*, 1(1):1–47, 1971. doi: 10.1080/0022250X.1971.9989787.

[165] Jonatan Gomez, Elizabeth Leon, Olfa Nasraoui, and Fabian Giraldo. The Parameter-less Randomized Gravitational Clustering algorithm with online clusters' structure characterization. *Progress in Artificial Intelligence*, 2(4):217–236, 2014. doi: 10.1007/s13748-014-0054-5.

[166] Yen-Jen Oyang, Chien-Yu Chen, and Tsui-Wei Yang. A Study on the Hierarchical Data Clustering Algorithm Based on Gravity Theory. In Luc De Raedt and Arno Siebes, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2168, pages 350–361. Springer, 2001. doi: 10.1007/3-540-44794-6_29.

[167] Chien-Yu Chen, Shien-Ching Hwang, and Yen-Jen Oyang. An incremental hierarchical data clustering algorithm based on gravity theory. In Ming-Syan Chen, Philip S. Yu, and Bing Liu, editors, *6th Pacific-Asia Conference, PAKDD 2002*, pages 237–250. Springer, 2002. doi: 10.1007/3-540-47887-6_23.

[168] Chien Yu Chen, Shien Ching Hwang, and Yen Jen Oyang. A statistics-based approach to control the quality of subclusters in incremental gravitational clustering. *Pattern Recognition*, 38(12):2256–2269, 2005. doi: 10.1016/j.patcog.2005.03.005.

[169] Ivan Gabrijel and Andrej Dobnikar. On-line identification and reconstruction of finite automata with generalized recurrent neural networks. *Neural Networks*, 16(1):101–120, 2003. doi: 10.1016/S0893-6080(02)00221-6.

[170] Jonatan Gomez, Juan Peña-Kaltekis, Nestor Romero-Leon, and Elizabeth Leon. INCRAIN: An Incremental Approach for the Gravitational Clustering. In Alexander Gelbukh and Ángel Fernando Kuri Morales, editors, *MICAI 2007: Advances in Artificial Intelligence*, pages 462–471. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-76631-5_44.

[171] Teng Long and LW Jin. A new simplified gravitational clustering method for multi-prototype learning based on minimum classification error training. In N. Zheng, X. Jiang, and X. Lan, editors, *International Workshop on Intelligent Computing in Pattern Analysis/Synthesis IWICPAS 2006*, pages 168–175, 2006. doi: 10.1007/11821045_18.

[172] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. GSA: A Gravitational Search Algorithm. *Information Sciences*, 179(13):2232–2248, 2009. doi: 10.1016/j.ins.2009.03.004.

[173] Melvin Gauci, Tony J. Dodd, and Roderich Groß. Why 'GSA: a gravitational search algorithm' is not genuinely based on the law of gravity. *Natural Computing*, 11(4):719–720, 2012. doi: 10.1007/s11047-012-9322-0.

[174] Mohammad Bagher Dowlatshahi and Hossein Nezamabadi-pour. GGSA: A Grouping Gravitational Search Algorithm for data clustering. *Engineering Applications of Artificial Intelligence*, 36:114–121, 2014. doi: 10.1016/j.engappai.2014.07.016.

[175] Chris Gorman and Iren Valova. GORMANN: Gravitationally Organized Related Mapping Artificial Neural Network. *The Computer Journal*, 2015. doi: 10.1093/comjnl/bxv069.

[176] Kun-Ta Chuang and Ming-Syan Chen. Clustering Categorical Data Using the Correlated-Force Ensemble. In *SIAM International Conference on Data Mining*, pages 269–278, 2004. doi: 10.1137/1.9781611972740.25.

[177] Jonatan Gomez and Elizabeth Leon. Spherical Randomized Gravitational Clustering. In *8th Alberto Mendelzon Workshop on Foundations of Data Management*, volume 2, 2014.

[178] Umut Orhan, Mahmut Hekim, and Turgay Ibrikci. Gravitational Fuzzy Clustering. In Alexander Gelbukh and Eduardo Morales, editors, *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pages 524–531. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-88636-5_50.

[179] Genyun Sun, Qinhuo Liu, Qiang Liu, Changyuan Ji, and Xiaowen Li. A novel approach for edge detection based on the theory of universal gravity. *Pattern Recognition*, 40(10):2766–2775, 2007. doi: 10.1016/j.patcog.2007.01.006.

[180] Shisheng Zhong, Xiaolong Xie, and Lin Lin. Two-layer random forests model for case reuse in case-based reasoning. *Expert Systems with Applications*, 42(24):9412–9425, 2015. doi: 10.1016/j.eswa.2015.08.005.

[181] Leonardo Enzo Brito da Silva and José Alfredo Ferreira Costa. A Gravitational Approach for Enhancing Cluster Visualization in Self-Organizing Maps. In *ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*, pages 48–54, 2013.

[182] Christian Wiwie, Jan Baumbach, and Richard Röttger. Comparing the performance of biomedical clustering methods. *Nature Methods*, 12(11): 1033–1038, 2015. doi: 10.1038/nmeth.3583.

[183] B Bergmann and G Hommel. Improvements of General Multiple Test Procedures for Redundant Systems of Hypotheses. In P Bauer, G Hommel, and E Sonnemann, editors, *Multiple Hypotheses Testing*, pages 100–115. Springer Berlin Heidelberg, 1988. doi: 10.1007/978-3-642-52307-6_8.

[184] S.S. Shapiro and M.B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 1965. doi: 10.1093/biomet/52.3-4.591.

[185] Nornadiah Mohd Razali and Yap Bee Wah. Power comparisons of Shapiro-Wilk , Kolmogorov-Smirnov , Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33, 2011.

[186] Howard Levene. Robust tests for equality of variances. In Ingram Olkin, Sudhish G. Ghurye, Wassily Hoeffding, William G. Madow, and Henry B. Mann, editors, *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, pages 278–292. Stanford University Press, 1960.

[187] Jyrko Correa-Morris. An indication of unification for different clustering approaches. *Pattern Recognition*, 46(9):2548–2561, 2013. doi: 10.1016/j.patcog.2013.02.016.

[188] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004. doi: 10.1145/990308.990313.

[189] R. Jenssen, J.C. Principe, D. Erdogmus, and T. Eltoft. The Cauchy-Schwarz Divergence and Parzen Windowing: Connections to Graph Theory and Mercer Kernels. *Journal of the Franklin Institute*, 6 (343):614–629, 2006.

[190] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[191] Joydeep Ghosh. Multiclassifier systems: back to the future. In F Roli and J Kittler, editors, *Proceeding of the Third International Workshop on Multiple Classifier Systems*, volume 2364, pages 1–15, 2002. doi: 10.1007/3-540-45428-4_1.

[192] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[193] Ana L.N. Fred and André Lourenço. Cluster Ensemble Methods: from Single Clusterings to Combined Solutions. In Oleg Okun and Giorgio Valentini, editors, *Supervised and Unsupervised Ensemble Methods and their Applications*, volume 126, pages 3–30. Springer Berlin Heidelberg, 2008.

[194] R. Ghaemi, M.N. Sulaiman, H. Ibrahim, and N. Mustapha. A Survey: Clustering Ensembles Techniques. In *International Conference on Computer, Electrical, and Systems Science, and Engineering*, volume 38, pages 644–653, 2009.

[195] Tao Li, Mitsunori Ogihara, and Sheng Ma. On combining multiple clusterings: an overview and a new perspective. *Applied Intelligence*, 33(2):207–219, 2010. doi: 10.1007/s10489-009-0160-4.

[196] Joydeep Ghosh and Ayan Acharya. Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315, 2011.

[197] Sandro Vega-Pons and José Ruiz-Shulcloper. A Survey of Clustering Ensemble Algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337–372, 2011. doi: 10.1142/S0218001411008683.

[198] Ana L.N. Fred. Finding Consistent Clusters in Data Partitions. In Josef Kittler and Fabio Roli, editors, *Proceedings of Second International Workshop on Multiple classifier systems*, pages 309–318. Springer Berlin Heidelberg, 2001. doi: 10.1007/3-540-48219-9_31.

[199] Ana L.N. Fred and Anil K. Jain. Data clustering using evidence accumulation. In *Object recognition supported by user interaction for service robots*, volume 4, pages 276–280. IEEE Comput. Soc, 2002. doi: 10.1109/ICPR.2002.1047450.

[200] Bernd Fischer and J.M. Buhmann. Bagging for path-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1411–1415, 2003. doi: 10.1109/TPAMI.2003.1240115.

[201] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus Clustering : A Resampling-Based Method for Class Discovery and Visualization of Gene. *Machine Learning*, 52:91–118, 2003. doi: 10.1023/A:1023949509487.

[202] Xiaoli Zhang Fern and Carla E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *International Conference on Machine Learning*, 2003.

[203] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*, page 36. ACM, 2004.

[204] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003. doi: 10.1093/bioinformatics/btg038.

[205] Ludmila I Kuncheva and Stefan T Hadjitodorov. Using diversity in cluster ensembles. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1214–1219. IEEE, 2004. doi: 10.1109/ICSMC.2004.1399790.

[206] Ana L.N. Fred and Anil K. Jain. Combining Multiple Clusterings Using Evidence Accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):835–850, 2005. doi: 10.1109/TPAMI.2005.113.

[207] Carlotta Domeniconi and Muna Al-Razgan. Weighted cluster ensembles. *ACM Transactions on Knowledge Discovery from Data*, 2(4):1–40, 2009. doi: 10.1145/1460797.1460800.

[208] Stefan T. Hadjitodorov, Ludmila I. Kuncheva, and Ludmila P. Todorova. Moderate diversity for better cluster ensembles. *Information Fusion*, 7(3):264–275, 2006. doi: 10.1016/j.inffus.2005.01.008.

[209] Alexander Topchy, Anil K. Jain, and William Punch. Clustering ensembles: models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1866–1881, 2005. doi: 10.1109/TPAMI.2005.237.

[210] Hanan G. Ayad and Mohamed S. Kamel. Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1): 160–173, 2008. doi: 10.1109/TPAMI.2007.1138.

[211] Natthakan Iam-On, Tossapon Boongoen, and Simon Garrett. LCE: a link-based cluster ensemble method for improved gene expression data analysis. *Bioinformatics*, 26(12):1513–1519, 2010. doi: 10.1093/bioinformatics/btq226.

[212] Jianhua Jia, Xuan Xiao, Bingxiang Liu, and Licheng Jiao. Bagging-based spectral clustering ensemble selection. *Pattern Recognition Letters*, 32(10):1456–1467, 2011. ISSN 01678655. doi: 10.1016/j.patrec.2011.04.008.

[213] F. Jorge F. Duarte, João M.M. Duarte, Ana L.N. Fred, and M. Fátima C. Rodrigues. Average cluster consistency for cluster ensemble selection. In Ana Fred, Jan L.G. Dietz, Kecheng Liu, and Joaquim Filipe, editors, *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, pages 133–148. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-19032-2_10.

[214] Sandro Vega-Pons, José Ruiz-Shulcloper, and Alejandro Guerra-Gandón. Weighted association based methods for the combination of heterogeneous partitions. *Pattern Recognition Letters*, 32(16):2163–2170, 2011. doi: 10.1016/j.patrec.2011.05.006.

[215] André Lourenço, Samuel Rota Bulò, Nicola Rebagliati, Ana L.N. Fred, Mário A.T. Figueiredo, and Marcello Pelillo. Probabilistic consensus clustering using evidence accumulation. *Machine Learning*, 98 (1-2):331–357, 2013. doi: 10.1007/s10994-013-5339-6.

[216] Zhi-Hua Zhou and Wei Tang. Clusterer ensemble. *Knowledge-Based Systems*, 19(1):77–83, 2006. doi: 10.1016/j.knosys.2005.11.003.

[217] Xiaoli Z. Fern and Wei Lin. Cluster Ensemble Selection. *Statistical Analysis and Data Mining*, 1(3): 128–141, 2008. doi: 10.1002/sam.10008.

[218] Javad Azimi and Xiaoli Fern. Adaptive cluster ensemble selection. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 992–997. Morgan Kaufmann Publishers Inc., 2009.

[219] Zhiwen Yu, Le Li, Yunjun Gao, Jane You, Jiming Liu, Hau-San Wong, and Guoqiang Han. Hybrid clustering solution selection strategy. *Pattern Recognition*, 47(10):3362–3375, 2014. doi: 10.1016/j.patcog.2014.04.005.

[220] Selim Mimaroglu and Emin Aksehirli. DICLENS: Divisive Clustering Ensemble with Automatic Cluster Number. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(2):408–420, 2012. doi: 10.1109/TCBB.2011.129.

[221] Xi Wang, Chunyu Yang, and Jie Zhou. Clustering aggregation by probability accumulation. *Pattern Recognition*, 42(5):668–675, 2009. doi: 10.1016/j.patcog.2008.09.013.

[222] André Ribeiro Lourenço. *Organizing Data Combining Multiple Sources Of Information*. PhD thesis, Universidade de Lisboa, Instituto Superior Técnico, 2014.

[223] André Lourenço, Samuel Rota Bulò, Ana L.N. Fred, and Marcello Pelillo. Consensus Clustering with Robust Evidence Accumulation. In Anders Heyden, Fredrik Kahl, Carl Olsson, Magnus Oskarsson, and Xue-Cheng Tai, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 307–320. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40395-8_23.

[224] Sandro Vega-Pons and José Ruiz-Shulcloper. Clustering Ensemble Method for Heterogeneous Partitions. In Eduardo Bayro-Corrochano and Jan-Olof Eklundh, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 14th Iberoamerican Conference on Pattern Recognition, CIARP 2009*, volume 5856 LNCS, pages 481–488. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-10268-4_56.

[225] Lucas Vendramin, Pablo A. Jaskowiak, and Ricardo J.G.B. Campello. On the combination of relative clustering validity criteria. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management - SSDBM*, pages 1–12. ACM Press, 2013. doi: 10.1145/2484838.2484844.

[226] M.C. Naldi, A.C.P.L.F. Carvalho, and R.J.G.B. Campello. Cluster ensemble selection based on relative validity indexes. *Data Mining and Knowledge Discovery*, 27(2):259–289, 2013. doi: 10.1007/s10618-012-0290-x.

[227] Raivo Kolde, Sven Laur, Priit Adler, and Jaak Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012. doi: 10.1093/bioinformatics/btr709.

[228] Isabelle Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[229] Isabelle Guyon, Gunn Steve, Nikravesh Masoud, and A. Zadeh Lotfi. *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, 2006. doi: 10.1007/978-3-540-35488-8.

[230] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28, 2014. doi: 10.1016/j.compeleceng.2013.11.024.

[231] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *Science and Information Conference*, pages 372–378. IEEE, 2014. doi: 10.1109/SAI.2014.6918213.

[232] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In Y Weiss, B Schölkopf, and J Platt, editors, *Advances in Neural Information Processing Systems*, pages 507–514. MIT Press, 2005.

[233] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 1151–1157. ACM Press, 2007. doi: 10.1145/1273496.1273641.

[234] Jung-Yi Jiang, Yao-Lung Su, and Shie-Jue Lee. MIKM: A mutual information-based K-medoids approach for feature selection. In *International Conference on Machine Learning and Cybernetics*, volume 1, pages 102–107. IEEE, 2011. doi: 10.1109/ICMLC.2011.6016694.

[235] Sam Roweis. EM Algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, volume 10, pages 626–632, 1997. doi: 10.1021/ja100409b.

[236] Michael E. Tipping and Christopher M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999. doi: 10.1111/1467-9868.00196.

[237] Claudio Ceruti, Simone Bassis, Alessandro Rozza, Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. DANCo: An intrinsic dimensionality estimator exploiting angle and norm concentration. *Pattern Recognition*, 47(8):2569–2581, 2014. doi: 10.1016/j.patcog.2014.02.013.

[238] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[239] Francesco Camastra and Antonino Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016. doi: 10.1016/j.ins.2015.08.029.

[240] Natthakan Iam-On and Tossapon Boongoen. Comparative study of matrix refinement approaches for ensemble clustering. *Machine Learning*, 98(1-2): 269–300, 2015. doi: 10.1007/s10994-013-5342-y.

[241] Richard McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Chapman & Hall/CRC Press, 2015. ISBN 9781482253443. URL `http://xcelab.net/rm/statistical-rethinking/`.

[242] Harold Jeffreys. An Invariant Form for the Prior Probability in Estimation Problems. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 186(1007):453–461, 1946. doi: 10.1098/rspa.1946.0056.

[243] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A. Brubaker, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, VV(Ii), 2016.

[244] Pablo A. Jaskowiak, Ricardo J.G.B. Campello, and Ivan G. Costa. On the selection of appropriate distances for gene expression data clustering. *BMC Bioinformatics*, 15(Suppl 2):S2, 2014. doi: 10.1186/1471-2105-15-S2-S2.

[245] Susmita Datta and Somnath Datta. Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics*, 7(1):397, 2006. doi: 10.1186/1471-2105-7-397.