



LOUVAIN
School of Management

Measures on Graphs: from Similarity to Density

Mathieu SENELLE

Thèse de doctorat 4 | 2014

Université catholique de Louvain

LOUVAIN SCHOOL OF MANAGEMENT RESEARCH INSTITUTE GROUP

Measures on Graphs: from Similarity to Density

Mathieu SENELLE 4 | 2014



Université catholique de Louvain
Louvain School of Management



Measures on Graphs: from Similarity to Density

Thesis submitted in partial fulfillment of the requirements for the
degree of Philosophiæ Doctor in Management Science

MATHIEU SENELLE

Jury Members:

Pr. François Fouss	Université catholique de Louvain, Promotor
Pr. Marco Saerens	Université catholique de Louvain
Pr. Masashi Shimbo	Nara Institute of Science and Technology, Japan
Pr. Fouad Riane	Faculté des Sciences et Techniques de Settat, Morocco
Dr. Amin Mantrach	Yahoo!, Spain
Pr. Claude Pecheux	Université catholique de Louvain, President

Mons, January 2014

Acknowledgements

During this thesis, I have met many different people who have all contributed to the realization of this work and to my personal development.

First of all, I am very grateful to my PhD. Advisor, François Fouss. “Chef”, thank you for your support throughout this thesis, your guidance and your availability. Your door was always (well, . . . often) open for my questions, or when I thought having discovered a revolutionary technique, and finally not. Most of all, thanks for your friendship.

A special thank goes to Marco Saerens, for his advice on what to do when I was in a dead end, his enthusiasm about science, mathematics (and about everything in general), his help on technical matters, and his human qualities.

Thanks also to the other members of the jury, Masashi Shimbo, Amin Mantrach and Fouad Riane for their involvement and their valuable suggestions, and Claude Pecheux who chaired this jury with brio. Masashi, thank you for having been our guide during our trip to Japan, which was a trip I will remember.

I would also like to thank all the people I met in FUCaM/UCL-Mons for the unique atmosphere prevailing there. My thoughts go to Virginie, Fabian, Benoît, Caro, Jean-Phi, Marie, Daisy, Sandrine, Geraldine, Thibaut, Judith, Orane, Natalie, Jean-Seb, Sabine, Karine, Sonia and so many more. Thanks for these years full of Friday drinks (with Francky Vincent), football matches (and discobus), processions to the Delhaize, epistemological torture, experiments on fruits and cheese, etc.

And last but not least, thanks to my family for supporting me all these years, and particularly Anne-Sophie who endured the end of my thesis with patience (you see, it has an end !), and Hélène whose smiles have enlightened those last months.

Table of contents

Table of contents	v
List of figures	ix
List of tables	xiii
General Introduction	1
Outline of the thesis	1
Managerial implications	2
Contributions	3
1 Preliminaries	5
I Similarity measures on graphs	9
2 Similarity measures between graphs	11
2.1 Introduction	12
2.2 Review of graph comparison methods	15
2.3 Proposed kernels	28
2.4 Experiments	32
2.5 Chapter conclusion	35

3	Bag-of-Forests similarities between nodes of a graph	37
3.1	Review of similarity measures on nodes of a graph	38
3.2	Introduction to the Matrix-Forest Theorem	47
3.3	A new forest similarity measure	49
3.4	Experiments	52
3.5	Chapter conclusion	68
II	Density measures on graphs	69
4	The Sum-over-Forests density index	71
4.1	Review of density measures on graphs	73
4.2	The SoF density index	77
4.3	Computation of the partition function $\mathcal{Z}$	78
4.4	Computation of the SoF density index	79
4.5	Experiments	80
4.6	Chapter conclusion	108
5	The Sum-over-Forests clustering on graphs	109
5.1	Review of density-based clustering	109
5.2	The SoF divisive clustering	112
5.3	The SoF mode-seeking clustering	114
5.4	Chapter conclusion	122
	General conclusion	125
	Main research results	125
	Applications	127
	Future prospects	128

Bibliography	129
Extended table of contents	145

List of Figures

1.1	From left to right: undirected, directed, and labelled directed graphs. . . .	5
1.2	Undirected, unweighted graph.	6
2.1	A molecule modelled with a graph G	12
2.2	An example of linearly separable problem, with a hyperplane separating two classes of objects (triangles and circles), and the corresponding margin (dashed lines).	19
2.3	An example of non-linearly separable problem, with the introduction of slack variables.	20
2.4	An example of non-linearly separable problem, illustrating the kernel trick. On the left are the original data in the input space, on the right are the data mapped into feature space, where they become linearly separable. . .	21
2.5	Extraction of the labelled sequences from graph G , and computation of the similarity between two sequences from graphs G and G'	24
2.6	Construction of the Shortest-Path Graph from initial graph G (a) and comparison of 1-Walks (b).	25
2.7	One iteration of the Weisfeiler-Lehman Kernel on a test graph.	27
2.8	Ranking of the features based on their importance in the classification, on four datasets (Nx A/L stands for Norm x of adjacency/Laplacian matrix). .	31
2.9	Characteristics of datasets.	32
2.10	Classification accuracy of the different kernels (in percents).	33
2.11	Computation time of the different kernels.	34
2.12	Correlation between the different kernels, on PTC dataset.	35

2.13	Correlation between the different kernels, on NCI1 dataset.	35
3.1	Diagram illustrating V , the set of nodes on the graph, and the neighborhoods of nodes i and k	39
3.2	Different neighborhood indicators defined as adjacency matrix elements of the corresponding graph.	39
3.3	A directed graph G in which arc costs are uniformly 1.	50
3.4	Examples of forests on graph G containing two trees.	50
3.5	Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 2 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).	56
3.6	Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 3 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).	57
3.7	Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 5 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).	58
3.8	Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (WebKB). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).	59
3.9	Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs. Results are reported for four methods (BoF, BoF+alpha, BoP with alignment (align), BoP with group betweenness (GBet)) and five labelling rates (10%, 30%, 50%, 70%, 90%).	67
4.1	3-communities (3C) datasets.	81
4.2	10-communities (10C) datasets.	81
4.3	S-Sets datasets.	82
4.4	NewsGroup datasets.	82

4.5	Grid dataset.	83
4.6	Correlation results for the 3-communities datasets, between true density and SoF density index, Strength, CC. Weighted (W) and Unweighted (U) graphs are considered, constructed with k-Nearest Neighbors (k-NN) or Threshold (Th) methods.	87
4.7	Correlation results for the 10-communities datasets, between true density and SoF density index, Strength, CC. Weighted (W) and Unweighted (U) graphs are considered, constructed with k-Nearest Neighbors (k-NN) or Threshold (Th) methods.	88
4.8	10-communities dataset (low sigma values S1), with SoF superimposed (Threshold 80, Unweighted) and having a negative correlation with the “true” density.	93
4.9	3-communities datasets for various σ values with the true density superimposed.	94
4.10	3-communities datasets for various σ values with the SoF density index superimposed.	94
4.11	10-communities dataset (low sigma values S1) with true density, SoF density index, and Strength superimposed (Threshold 95, Weighted).	95
4.12	10-communities dataset (low sigma values S1) with true density, SoF density index, and Strength superimposed (Threshold 90, Weighted).	95
4.13	10-communities dataset (high sigma values S2) with true density, SoF density index, and Strength superimposed (Threshold 95, Weighted).	95
4.14	10-communities dataset (high sigma values S2) with true density, SoF density index, and Strength superimposed (Threshold 90, Weighted).	96
4.15	S-Sets datasets with SoF density index superimposed (Threshold 95, Weighted).	96
4.16	S-Sets datasets with strength superimposed (Threshold 95, Weighted).	96
4.17	S-Sets datasets with CC superimposed (Threshold 95, Weighted).	97
4.18	NewsGroup datasets with SoF density index superimposed (Threshold 95, Weighted).	97
4.19	Grid dataset with SoF density index superimposed (Threshold 95, Weighted).	98

4.20	Generalized k -Core curve for 10C-S1, constructed with 90-percentile threshold method (a) and k -Nearest Neighbors with $k = 50$ (b). X-axis represents the iteration number, while the Y-axis represents the value of the true mean density.	100
4.21	Generalized k -Core curve for 10C-S2, constructed with 90-percentile threshold method (a) and k -Nearest Neighbors with $k = 50$ (b). X-axis represents the iteration number, while the Y-axis represents the value of the true mean density.	101
4.22	Different iterations of the k -Core procedure used to identify dense regions on 10C-S2.	102
4.23	10-communities datasets (high sigma values S2) with Strength and SoF density index superimposed (Threshold 90, Weighted).	104
4.24	NewsGroup1 datasets with Strength and SoF density index superimposed (Threshold 90, Weighted).	105
4.25	SoF density index distribution on the nodes of the 10-communities dataset (high sigma values S2) (Threshold 90, Weighted).	106
4.26	SoF density index distribution on the nodes of the 10-communities dataset (high sigma values S2) (Nearest Neighbors 50, Weighted).	107
5.1	SoF divisive clustering on 10C-S1, constructed with 95-percentile threshold method, and SoF's parameter $\theta = 10$	115
5.2	SoF clustering on 10C-S1 and 10C-S2, constructed with 95-percentile threshold method, and SoF's parameter $\theta = 10$. Each cluster is represented by a color.	116
5.3	S-Set S4 dataset with SoF density index superimposed. (a) represents the original SoF density index, (b) the density index divided by its smoothed version, (c) the original density index in 3D, and (d) the modified density index in 3D. Z-axis for (c) and (d) represents the magnitude of the density indices.	117
5.4	SoF clustering on S-Set S4 dataset, constructed with 95-percentile threshold method, and SoF's parameter $\theta = 10$. Each cluster is represented by a color.	118

List of Tables

1.1	Adjacency matrix corresponding to graph in Figure 1.2.	6
3.1	Composition of the nine Newsgroup datasets.	53
3.2	Composition of the WebKB cocite datasets.	53
3.3	One-side t -test for every labelling rate for NewsGroup datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.	60
3.4	One-side t -test for every labelling rate for WebKB datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.	61
3.5	One-side t -test for every labelling rate for all datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.	62
3.6	Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on NewsGroup datasets.	63
3.7	Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on NewsGroup datasets.	63
3.8	Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on NewsGroup datasets.	63
3.9	Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on WebKB datasets.	64
3.10	Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on WebKB datasets.	64

3.11	Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on WebKB datasets.	64
3.12	Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on all datasets.	65
3.13	Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on all datasets.	65
3.14	Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on all datasets.	65
3.15	Ranking of the different methods using a Borda ranking method on News-Group datasets and for all labelling rate.	66
3.16	Ranking of the different methods using a Borda ranking method on WebKB datasets and for all labelling rate.	66
3.17	Ranking of the different methods using a Borda ranking method on all datasets and for all labelling rate.	66
4.1	(σ_x, σ_y) (standard deviations) values for the 10-communities datasets, for two degrees of overlapping between the clusters (S1 small overlapping, S2 strong overlapping).	82
4.2	Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.05$. . .	89
4.3	Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.1$. . .	89
4.4	Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.5$. . .	90
4.5	Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 10C-S1 dataset.	90
4.6	Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 10C-S2 dataset.	91
4.7	Spearman’s correlations between the Strength (Str), the smoothed version of Str with Laplacian smoothing (StrLap), and the “true density” on 10C-S2 dataset.	91

4.8	Spearman’s correlations between the Strength (Str), the smoothed version of Str with Page-Rank smoothing (StrPR), and the “true density” on 10C-S2 dataset.	92
4.9	Spearman’s correlations between the clustering coefficient (CC), the smoothed version of CC with Laplacian smoothing (CCLap), and the “true density” on 10C-S2 dataset.	92
4.10	Spearman’s correlations between the clustering coefficient (CC), the smoothed version of CC with Page-Rank smoothing (CCPR), and the “true density” on 10C-S2 dataset.	93
4.11	“Area” score for the generalized k -core procedure on 10C-S1 and 10C-S2. .	99
4.12	Skewness absolute values for the SoF density index distribution on the nodes of the 10-communities datasets (high sigma values S2), for different θ values.	103
5.1	Characteristics of the five networks used in clustering task.	120
5.2	Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) values for five networks. SoF+nbClust represents the SoF clustering method with number of clusters in input. When the number of clusters is not known (SoF and Louvain), the number of clusters found are given in parenthesis. .	122

General Introduction

Outline of the thesis

In this thesis, we mainly consider two closely related classical and general problems in data mining and machine learning: (1) developing new similarity measures between graphs and between nodes of a graph, and (2) developing new density measures on nodes of a graph.

The problem of comparing graphs finds its origin in the graph isomorphism problem, which is defined as determining if two graphs are isomorphic, or in other words, if they are equivalent up to a reordering of the nodes. It belongs to *NP* problems, but no proof has been given that it is either *P* or *NP-complete*. However, best known algorithms run in exponential time, except for special classes of graphs [7]. Besides this exact matching problem, other classes of similarity measures for graphs were developed in order to overcome this complexity limitation, and particularly one which has received a lot of attention recently: graph kernels. In the first part of this thesis, we introduce three new kernels which are derived from classical graphs kernels, like the shortest-path kernel from Borgwardt [15]. We then assess these kernels on a graph classification task in the field of bioinformatics and chemoinformatics, using molecules from various origins as graphs and trying to classify them according to one of their property (mutagenicity,...).

We then study the problem of quantifying the similarity between nodes of graphs. From the matrix-forest theorem developed by Chebotarev [24], which led to a “forest similarity” measure, we define a new measure by assigning the forests present in the graph a probability of being sampled at random, favoring small (low-cost) forests against bigger ones. This measure is called the Bag-of-Forests (BoF) similarity measure, from which we also derive two other distances: the BoF potential distance and the BoF logarithmic distance, which can be turned back again into similarities. Those new measures are then assessed on a semi-supervised classification task, trying to infer missing node labels from the ones already present in the test datasets.

While developing the BoF similarity measure, we had the intuition that this similarity could be extended into another class of measure: a density measure on graphs. We then developed the Sum-over-Forests (SoF) density index in the same spirit than the BoF similarity, guessing that when picking forests which are assigned a probability of

sampling, if a node is present in many (preferably short) forests, it should lie in a dense region of the graph. This density index can be computed for all nodes on the graph, as a density score, revealing the dense regions on graphs. This SoF index is then assessed in various ways: (1) in a visual way, superimposing density scores as colors (depending on the density values) on graph plots, (2) computing a correlation between those scores and the “true” density, which is known for artificial datasets, and (3) using a generalized k -core procedure [12], removing nodes presenting a low score, until convergence to the densest regions.

At last, from the SoF density score, we developed two clustering algorithms, which are able to work directly on graphs and where the number of clusters do not need to be specified a priori. The first one is a divisive algorithm, starting the clustering procedure with the whole graph, removing nodes with the lowest SoF density scores (in a k -core way) iteratively, revealing step by step the underlying clusters. The second one uses a mode-seeking procedure [80], identifying the modes of the graphs (which are peaks of density) as peaks of SoF density score. In those two algorithms, the clusters are then formed assigning each node to its nearest cluster, using for instance the BoF similarity measure to perform this assignment.

It should be noted that only graphs of moderate size (up to 5000 nodes) are used in this work, as the processing of larger graphs requires the set up of faster variants of the methods developed here, which is left for future work. Moreover, all the experiments performed here were implemented in Matlab, with additional help of some toolboxes, as LibSVM [19] for the classification of graphs.

Managerial implications

The managerial application of the similarity between graphs is discussed in Chapter 2. We show how the modelling of a molecule as a graph, and its use in automatic classification could lead to enormous savings for pharmaceutical companies when developing new drugs. Indeed, candidate molecules can be “screened” virtually (i.e., via a computer) and no more via chemical experiments, which enables to check the candidates molecules in a much faster way. Similarly, in natural language processing, documents can be modelled as graphs and classified according to their content [3, 137].

The applications of the similarity between nodes of a graph are numerous [3]. We applied our new measures to a semi-supervised classification task, aiming at identifying unlabelled nodes from the information beared by labelled nodes. As an example, this can be applied to industrial inspection applications with machine vision systems in order to improve classification accuracy, by making use of the information contained in the unlabeled

belled samples and thus reduce the labeling burden of a human operator [165]. Another application of the use of nodes similarity is the collaborative recommendation [51], which proposes different products (like books, movies, ...) to users, both modelled as nodes of a graph. A similarity measure is then needed to find, for instance, the consumers which are the closest to each other: if a person likes certain movies, another person which is close is likely to enjoy those movies too. This kind of recommendation system (or variants) is used by web sites like Amazon or Netflix (which proposes movies) [110].

Finding dense regions on graphs is an important subject in bioinformatics. Actually, protein-protein interaction networks are graphs which model the way proteins interact with each other, which is related to their specific function. Finding dense components in those networks thus helps to explain metabolic processes and to label proteins whose function are unknown [3]. In social networks, a dense region may indicate a community structure, a cohesive set of people sharing similar education, living zone, tastes, ..., which can be useful in terms of marketing, for instance.

At last, clustering nodes of a graph appears in many areas [136]. On a telecommunication network, such as a mobile phone network, one possible clustering is to group users based on their language [14]. In the marketing field, cluster analysis can be used in market segmentation to partition the consumers into market segments, which can be useful when sending specific advertisements. In image segmentation, images can be modelled as graphs, and clustering methods are able to distinguish different objects in those images [44], for instance being able to separate human faces from the background.

Contributions

As already stated in the previous section, this thesis is centered on the study of four problems, for which we summarize our contributions to their respective fields:

- Similarity measures between graphs: from the class of methods called graph kernels, we derived **three new kernels**, trying to improve either accuracy or computation time compared to state-of-the-art kernels. We published this work as a *UCL-LSM Working Paper* [143].
- Similarity measures between nodes of a graph: from the matrix-forest theorem [24], we derived **three new “forest similarity” measures** known as the Bag-of-Forests (BoF) similarity, the BoF potential similarity and the BoF logarithmic similarity, and applied them to a semi-supervised classification task.
- Identifying dense regions on graphs: few density measures exist on graphs, and are often derived from clique (or other dense subgraphs) searching, which is time

consuming. Continuing on the same track than the BoF similarities, we developed the **Sum-over-Forests (SoF) density index**, which assigns a density score to the nodes of a graph. A paper has been written on this density index and has been accepted for publication in *IEEE TPAMI* [142].

- Clustering on graphs: as the SoF density index identifies dense areas on graphs, we created **two density-based clustering methods**. Those two methods work directly on graphs, and do not need to be provided with the number of clusters a priori, contrarily to some classical methods, like K-Means. A paper on this subject was published as a *UCL-LSM Working Paper* [144], and has also been submitted in a conference.

Chapter 1

Preliminaries

Graphs can be found in many fields, and are able to represent many real life objects, like molecules, social networks, telecommunication networks,... But what do these graphs really are ? In fact, they can be defined as mathematical objects containing nodes (or vertices) and edges. In Figure 1.1, nodes are modelled by circles and edges by links going from one node to another. There are several kind of graphs: labelled or not, directed or not. Labelled graphs bear information (labels) on their nodes and/or on their edges (when the labels belong to an ordered set (for instance the real numbers), it is also called a weighted graph). Directed graphs have edges that are directed, i.e., they have a direction assigned to them.

Mathematically speaking (see, for instance [35, 157]), a graph G is a pair $G = (V, E)$, where E is the set of *edges*, characterized by a couple (v_i, v_j) where v_i, v_j are two nodes belonging to V , the set of nodes. Those nodes are linked by the edge $e_{i,j}$. The size of the graph G is defined as the cardinality of the set V . An undirected graph is a graph where edges $e_{i,j}$ are equal to $e_{j,i}$, $\forall i, j$. It is directed if $e_{i,j}$ is different from $e_{j,i}$. A loop is an edge where the starting node is the same as the ending node, namely $e_{i,i}$. A graph without loops is called a simple graph. A graph is called weighted (or labelled) if nodes and/or edges bear weights, which often represent an information of affinity between the nodes (or inversely a distance), or an information from the domain where the graphs comes (as

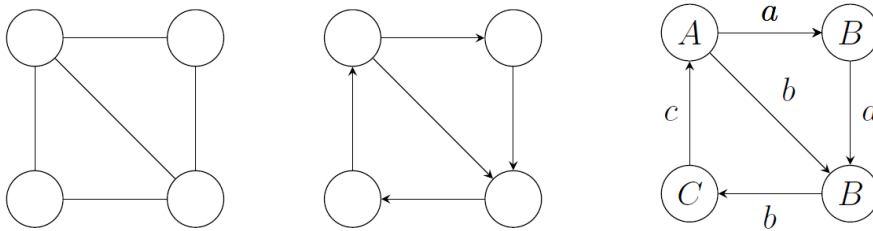


Figure 1.1: From left to right: undirected, directed, and labelled directed graphs.

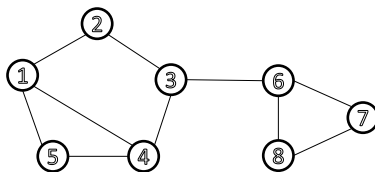


Figure 1.2: Undirected, unweighted graph.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Table 1.1: Adjacency matrix corresponding to graph in Figure 1.2.

the bond in molecules).

A graph G can be described by its adjacency matrix $\mathbf{A}$ (in the rest of the document, matrices will be represented with a bold uppercase letter) which, in the unweighted case, is a square binary matrix where rows and columns represent nodes. All its elements a_{ij} are set to 0, except when there is a connection between two nodes: a 1 value is put at the intersection of the row corresponding to the starting node of the edge, and the column corresponding to the ending node of the edge. In the weighted case, the 1 values are multiplied by the weights of the corresponding edges. In an undirected graph, the set of nodes directly connected to a node i is called the “neighborhood” of node i , and is denoted $\mathcal{N}(i)$. The “successors” of a node i in a directed graph are the nodes to which edges coming from i arrive, while “predecessors” are the starting nodes of edges arriving in i . An example of an undirected, unlabelled graph is shown in Figure 1.2, and its corresponding adjacency matrix in Figure 1.1. The adjacency matrix of an undirected graph is symmetric, i.e., $\mathbf{A} = \mathbf{A}^T$, where $\mathbf{A}^T$ is the transpose of matrix $\mathbf{A}$.

A *subgraph* of G is a graph whose edges and vertices belong to V and E . A *spanning subgraph* of G is a subgraph of G whose vertex set is the same as the one of G . A *multigraph* is a graph which may contain more than one edge between two vertices. A *walk* is a sequence of nodes and edges, and a *path* is a walk where no vertex is repeated in the sequence (no cycles are permitted). The *shortest path* between two nodes is the path where the sum of the weights on the edges constituting this path is the minimal one, among all paths between these two nodes. On unweighted graphs, the shortest path is simply defined as the path with the minimum number of edges. A *tree* is a graph where

two vertices are connected by exactly one path, thus allowing no cycle. A *forest* is a disjoint union of trees. A *rooted forest* is a forest where each of its components (trees) is rooted. A *rooted tree* is a tree with one vertex which is marked, and designated as the root.

On each arc linking nodes i and j , we can associate a positive number $c_{ij} > 0$ representing the **immediate cost** of following this arc. The **cost matrix** $\mathbf{C}$ is the matrix containing the immediate costs c_{ij} as elements. If, instead of $\mathbf{C}$, we are given an adjacency matrix with elements $a_{ij} \geq 0$ indicating the affinity between node i and node j , the corresponding costs could be computed from $c_{ij} = 1/a_{ij}$. Notice, however, that other relations between affinity and cost could be considered as well. The Laplacian matrix of a graph having adjacency matrix $\mathbf{A}$ is $\mathbf{L}(\mathbf{A}) = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathbf{Diag}(\mathbf{A}^T \mathbf{e})$ is a diagonal matrix containing the column sums of $\mathbf{A}$. Here, $\mathbf{e}$ is a column vector full of 1's.

The exponential and logarithm operations on matrices used in this work are element-wise operations. For instance, the exponential of matrix $\mathbf{A}$ is a new matrix containing the exponential of each a_{ij} :

$$\exp(\mathbf{A}) = \begin{pmatrix} \exp(a_{11}) & \cdots & \exp(a_{1m}) \\ \exp(a_{21}) & \cdots & \exp(a_{2m}) \\ \vdots & \ddots & \vdots \\ \exp(a_{m1}) & \cdots & \exp(a_{mm}) \end{pmatrix}.$$

Part I

Similarity measures on graphs

Chapter 2

Similarity measures between graphs

Contents

2.1	Introduction	12
2.1.1	Overview of the research topic	12
2.1.2	Applications	13
2.2	Review of graph comparison methods	15
2.2.1	Graph isomorphism	15
2.2.2	Graph-Edit distance	15
2.2.3	Graph features	16
2.2.4	Frequent subgraphs	17
2.2.5	Graph kernels	17
2.3	Proposed kernels	28
2.3.1	Modified Weisfeiler-Lehmann kernel	28
2.3.2	Randomized Shortest-Path kernel	28
2.3.3	Graph Features kernel	29
2.4	Experiments	32
2.4.1	Datasets	32
2.4.2	Experimental setup	33
2.4.3	Results	33
2.5	Chapter conclusion	35

2.1 Introduction

2.1.1 Overview of the research topic

Bioinformatics, chemoinformatics, drug discovery are only few examples of domains where the need for algorithms to analyze and classify data is constantly increasing. Indeed, various applications, like predicting the properties of molecules, such as toxicity or anti-cancer activity, are using databases with millions of molecules to handle [146]. An efficient way to model these molecules is to use structured objects like graphs, because in these cases, we need to take into account not only their sub-elements (atoms, which are represented as nodes), but also the structure between them (bonds, which are represented as edges). We can see an example of molecule modelled by a graph on the figure 2.1.

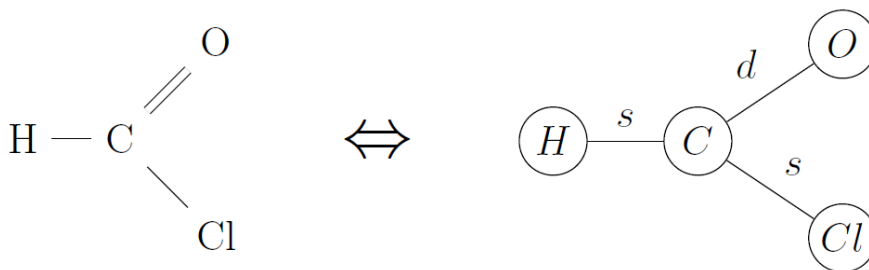


Figure 2.1: A molecule modelled with a graph G .

In this context, two problems can be studied: firstly, we can try to find how similar two nodes in a graph are, and secondly, how similar two graphs, or a set of graphs, are.

In this first part of this work, we will study different ways to answer the second question. We will focus on finding and analyzing new similarity measures between graphs, which are efficient in a computational point of view (i.e., the time to solve a problem (here processing graphs) should grow as slowly as possible when its size increases), but also lead to accurate results, compared to state-of-the-art methods. This work will also show that the suggested measures can be applied successfully to supervised classification, which will let us assess the quality of those measures.

In the second part of this thesis, we will focus on finding new similarity measures on the nodes of a graph, and derive a way to assess density on nodes. Clustering algorithms based on this density score will also be introduced in the second part of this work.

2.1.2 Applications

The discovery of new drugs is a long and expensive process. Indeed, it takes 10 to 15 years to develop a drug while the average cost associated to it is about 800 million euro [36]. Pharmaceutical companies therefore try to diminish these costs, and the use of machine learning and data mining tools become prominent to achieve that goal. In this section, we will detail the traditional way used to discover new drugs, and how computational techniques can improve this process.

The primary goal of new drugs research is to identify the origin of the apparition of a disease, and this factor is usually an active protein (called the target)[37]. We thus try to find a compound which will interact with that protein, and will inhibit its activity. Of course, that compound must not only be active towards the protein, but also be harmless for patients who will ingest the corresponding drug. We will now develop each step of the process leading to the discovery of a new drug. The first step of target identification involves deep knowledge in biology and chemistry. We will not go further in this subject, as it is not impacted with bioinformatics techniques used in our research, but, as an example, the work realized on the Human Genome Project, which aimed at decrypting the complete DNA of the human genome, permitted the identification of many new targets [149].

The second step of the discovery process is to find a molecular compound which will bind to the target defined in the first step. But where do these compounds come from? In fact, during the eighties, a technique called combinatorial chemistry was set up. It uses a set of elementary chemical blocks, and combines them in every possible way [164]. This operation allows scientists to synthesize huge sets of molecules. For example, if we consider 20 building blocks (for instance amino acids), and we would like to synthesize molecules constituted by 5 blocks, we can therefore create $20^5 = 3200000$ molecules ! Thereby, we can then obtain sets of millions molecules, which will have to be screened to determine the ones that present the best characteristics (i.e., interaction with the target). The screening step consists in sending the molecule studied on the target, see if it binds to it, and to quantify the strength of the “binding” [67]. We can imagine the time it would take to screen millions of molecules “by hand”. Fortunately, a technique allowing to screen molecules in a very fast way exists: the high throughput screening (HTS), consisting in a parallelization of the screening. As an order of magnitude, the latest version of HTS is able to treat 100 000 molecules a day [91]. The third step of the discovery process is to verify that candidate molecules, which are an output of the HTS step, possess the adequate characteristics to be used as a treatment to human patient. Indeed, not only these molecules have to interact with the target, but they also have to possess low toxicity and good chemical properties (ability to enter the bloodstream, to be eliminated from the body,...)[102, 171].

Let us now describe how computational biology can intervene and improve the drug discovery process described above. First of all, the sets of molecular compounds created by combinatorial chemistry are not much diversified. Indeed, a lot of molecules present almost the same structure, and it is known that a similar structure is correlated to a similar function [40]. So if the screening is performed on all molecules, a lot of redundancy will be found. A way to avoid this is to group the similar molecules together, by creating clusters. This implies to define a similarity measure between the compounds, which is exactly the goal of our work. We will represent molecules as graphs, and check their similarity with a measure we will set up. As the database is constituted with the graphs representing the molecules, we will then have clusters that regroup similar molecules [105]. The screening step will therefore be simplified, as we will only have to check a representative molecule of a cluster, and if this molecule does not bind well to the target, we will drop that cluster and check another. If the interaction between the molecule and the target is strong enough, then the other molecules belonging to the cluster will also be screened, in order to determine the best candidate for a drug. This process is known as Library Design [171]. Another way to improve the drug discovery cycle using computational tools is the following: when the structure (the graph representation) of a reference molecule, which is known to bind strongly to the target, is accessible, we can also perform what is called a Virtual Screening [8]. The principle is quite simple: we compare molecules present in the database to the one of reference. Again, as similar structure gives similar effects, we will first “virtually” (i.e., on a computer) select the molecules the most similar to the reference, and then perform the “physical” screening only on that restrained pool of molecule [67]. This is an enormous time saving, compared to the case where all the molecules have to be “physically” screened.

We can see that our work fits well in a research area which is very interesting for pharmaceutical companies. Indeed, by performing the operations above, the huge sets of molecules created by combinatorial chemistry are restrained to a set of good candidates, and the “physical” screening is only required on this smaller set of compounds. Our main contribution in the context of drug discovery is to improve existing similarity measures between graphs (representing molecules) and to create new ones. Ideally, a measure of similarity should meet two conditions: firstly it should be computationally efficient, because it is important when handling large molecular databases, and secondly the measure of similarity should separate molecules considering the activity at interest. Finding such similarity measures is one of the current challenges in chemoinformatics and bioinformatics, and any improvement in this domain could lead to enormous savings for pharmaceutical companies.

2.2 Review of graph comparison methods

The methods used to compare graphs can be grouped in several categories. The most important of them are reviewed hereunder.

2.2.1 Graph isomorphism

One of the most intuitive idea when trying to assess how similar two graphs are is to determine if they are identical (in a topological point of view), or in other words if they are *isomorphic*. More formally, two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are isomorphic if there exists a bijection $f : V_1 \rightarrow V_2$ such that for all $(v_1^i, v_1^j) \in V_1 : (v_1^i, v_1^j) \in E_1 \Leftrightarrow (f(v_1^i), f(v_1^j)) \in E_2$. In other words, G_1 and G_2 are isomorphic if there exists a rearrangement of the nodes, without changing any edges, such that the two graphs are identical. The complexity of the graph isomorphism problem is one of the major open problems, as it is proven to be *NP*, but no proof exists to either determine if it is *P* or *NP-complete* [128]. The existing algorithms have however an exponential time complexity in the worst case [46]. Variants of this problem use subgraph isomorphism, where two graphs can be of different sizes. The problem is then to determine if the smallest graph exists up to an isomorphism in the biggest one (i.e., is isomorphic to a subgraph in the biggest graph). The subgraph isomorphism problem has been proven to be *NP-complete* [54].

2.2.2 Graph-Edit distance

Graph similarity measures based on isomorphism search for an exact matching between two graphs (or subgraphs), which can be overly restrictive: if an edge is missing, or has been deleted by error, two graphs which are almost similar will receive a similarity measure of 0. Other kinds of methods, like Graph-Edit distance [114, 135] are based on an inexact matching between graphs. Edit distance was first applied between strings of characters: the minimum edit distance between two strings is the minimum number of edition operations (like insertion, deletion, substitution of characters) to transform one string into the other [162]. Methods based on graph-edit distances use a similar principle: two graphs G_1 and G_2 are similar if G_1 can be transformed into G_2 (or inversely) using simple operations such as addition or deletion of nodes, and relabeling of nodes or edges in the case of labelled graphs. The cost for using one of these operations may vary from one operation to another. The graph-edit distance between G_1 and G_2 is then defined as the minimum cost of transforming G_1 into G_2 . However, the graph-edit distance requires the adequate definition of the cost function, which can be difficult to tune. An automatic determination of cost function has yet been developed [113], but the complexity of the

graph-edit distance algorithms remains exponential.

2.2.3 Graph features

Feature based methods extract several topological indicators from a graph, and use them to construct a feature vector which describes that graph. Those indicators can be computed from the structure of the graph (like number of nodes, number of edges,...) or can be obtained from the application studied, for instance biological activity of molecules. This kind of graph comparison is widely used in bio or chemoinformatics. Some of the most widely used descriptors on undirected graphs are presented below: the Wiener index, the Morgan index, the Zagreb index and the Randic index.

2.2.3.1 The Wiener index [170]

The Wiener index is defined as the sum of all the shortest paths on a graph :

$$W(G) = \sum_i \sum_j D_{ij}, \quad (2.1)$$

where D_{ij} is the shortest path between nodes v_i and v_j .

2.2.3.2 The Morgan index [111]

The Morgan index of order k for node v is iteratively defined as

$$M(G, v) = \begin{cases} 1 & \text{if } k = 0 \\ \sum_{w \in \mathcal{N}(v)} M_{k-1}(w) & \text{otherwise.} \end{cases} \quad (2.2)$$

This index can also be seen as the number of walks of k steps starting at v in G .

2.2.3.3 The Zagreb index [59]

The Zagreb index can in fact be decomposed into two indices:

$$Z_1(G) = \sum_{v \in V} d^2(v), \quad (2.3)$$

$$Z_2(G) = \sum_{(v,w) \in E} d(v)d(w), \quad (2.4)$$

where $d(v)$ and $d(w)$ are the degrees of node v and w .

2.2.3.4 The Randic index [126]

The Randic index is defined by:

$$R(G) = \sum_{(v,w) \in E} \frac{1}{d_v d_w}, \quad (2.5)$$

where d_v and d_w are also the degrees of nodes v and w .

Number of other descriptors exists, and this may be the principal drawback of this kind of methods: the most discriminant set of descriptors often vary from one application to another, and their computational complexity can be quite high (often polynomial or even exponential).

2.2.4 Frequent subgraphs

Frequent subgraphs comparison [57] consists in retrieving the most frequent subgraphs appearing in a graph, and comparing that collection of subgraphs to the ones from other graphs in the dataset. One of the most well-known algorithm used on this kind of tasks, the APRIORI algorithm is based on the *a priori* hypothesis [4], namely, a set of items are frequent only if all of the subsets it contains are also frequent. Equivalently, if an itemset is not frequent, no superset of it can be frequent. Using this information allows to reduce the search space in an efficient way when searching for the frequent itemsets, as the number of candidates can be reduced. [172] introduced a new algorithm called gSpan, which is based on a lexicographic ordering system and a depth-first search-based for efficient mining of frequent subgraphs, which outmatch the APRIORI algorithm on large datasets. Some methods also combine pattern mining and learning, like gBoost [133], which is a boosting method for graph data that uses linear programming and column generation techniques and CORK [155], which makes use of a quality criterion that allows to select frequent subgraphs in a greedy manner. The main drawback of these methods is that they are plagued by computational slowness, as they can scale exponentially with the size of graphs. They indeed have to enumerate all subgraphs of a graph at worst, which is unmanageable when graphs have more than a few nodes.

2.2.5 Graph kernels

Kernel methods are one of the latest - and most promising - technique used in data mining and machine learning to quantify similarities between structured objects. Unlike some classical methods which define a vector-based representation of the graphs, which can be tedious to construct, kernel methods do not represent data individually, but through

a set of pairwise comparison. More formally, suppose that we have a dataset X where each object x_i is, for instance, a molecule. In feature-based methods, which are often used in chemoinformatics, a representation $\phi(x_i)$ must be found for each object, leading to the set of representations $\phi(X)$. For a molecule, this representation could be the types of atoms present in that compound and thus $\phi(X) = (COO, HCl, \dots)$. As already said, kernel methods are based on a drastically different idea, i.e., the whole dataset is simply represented as a kernel matrix K , with $K_{i,j} = k(x_i, x_j)$ k being a similarity measure between i and j objects (see next section). This kind of representation has several advantages. Firstly, no matter the nature of the objects studied, the dataset will always be described as a real-valued matrix. As a consequence, an algorithm designed to handle such matrices will be able to process molecules as well as pictures. Secondly, it is often easier to find a comparison “score” between two complex objects than a exact representation of the latter. And thirdly, the size of the kernel matrix is only a function of the number of objects, and does not grow with their complexity.

2.2.5.1 Classification and Support Vector Machines

Support Vector Machines (SVMs) belong to the family of kernel methods, and are used in this work to perform supervised classification of molecules (see for instance [28, 30, 145, 159] for a thorough treatment of the subject). Classification is a data mining technique used to predict which class an object belongs to, and the term supervised is used in situations where each instance of the data consists of a pair composed of an input value and a desired output value. SVMs are often employed in binary classification problems: given a set of training objects $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$, $d \in \mathbb{N}$ associated with class labels $y_i \in \mathcal{Y} = \{\pm 1\}$ the goal is to build a classifier function $f : \mathcal{X} \rightarrow \mathcal{Y}$ in order to predict the label of any unknown object presented to that classifier.

Linearly separable problems

When data can be linearly separated into two classes, classification can be done by introducing a hyperplane (or a plane, if the data lie in three dimensions) between the classes $y = 1$ and $y = -1$ (see Figure 2.2). The class corresponding to an object $\mathbf{x}_i$ is determined by its location with respect to the hyperplane. If a dataset is linearly separable, an infinity of hyperplanes correctly classify the data, and are described by the equation $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, where $\mathbf{w}$ is a vector perpendicular to the hyperplane (and thus is an indicator of the direction of the plane), and b is a constant $\in \mathbb{R}$. These hyperplanes correspond to the following decision function :

$$f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b), \quad (2.6)$$

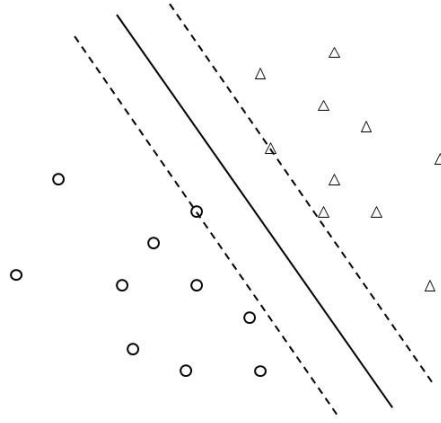


Figure 2.2: An example of linearly separable problem, with a hyperplane separating two classes of objects (triangles and circles), and the corresponding margin (dashed lines).

where $f(\mathbf{x})$ correspond to the class predicted for an object $\mathbf{x}$, i.e., $+1$ or -1 . But among all these hyperplanes, how to choose the “best” one ? One way to do that is to define a *margin*, which is the minimum distance between the hyperplane and the nearest points from both classes, and to select the hyperplane which maximizes that margin. It will be the optimal hyperplane, with respect to the margin criterion, as it minimizes the risk of misclassification of data. This optimization problem can be formulated as follow :

$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w}\|^2 \quad (2.7)$$

$$\text{subject to } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \text{ for all } i \in \{1, 2, \dots, m\}. \quad (2.8)$$

The minimization of $\mathbf{w}$ correspond to the objective of maximizing the margin. Indeed, it can be shown that the latter is proportional to $\left\| \frac{1}{\mathbf{w}} \right\|$. The constraints $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ illustrates the fact that any object must lie outside the margin.

Using the Lagrangian, we can solve the corresponding dual problem :

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad (2.9)$$

$$\text{subject to } \sum_{i=1}^m \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0 \text{ for all } i \in 1, 2, \dots, m, \quad (2.10)$$

and we also obtain :

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i. \quad (2.11)$$

One interesting fact about these equations is that the dual problem only take into account the dot product of the initial data set, it does not need the data in itself. This kind of Support Vector Machines is called *hard margin* SVM, because by construction, it does not tolerate an object to be misclassified.

Non-linearly separable problems

Unfortunately, the majority of the datasets are hardly linearly separable, and it is often impossible to find a hyperplane that correctly separate the objects in two classes. In order to overcome this problem, *soft margin* SVMs were developed. They allow some data to be misclassified, in order to improve the generalization performance of the SVM (the performance over previously unseen data)(see Figure 2.3). In 1995, Cortes and Vapnik [28] proposed a technique which tolerates misclassification, called C-SVM. It aims at minimizing the number of errors by introducing new slack variables, ξ_i , and a penalty term that penalizes high slack variables, C . The formulation of the primal problem now becomes :

$$\underset{\mathbf{w}, b, \xi}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \quad (2.12)$$

$$\text{subject to } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ for all } i \in \{1, 2, \dots, m\}. \quad (2.13)$$

C is a constant that controls the tradeoff between classification errors and the width of the margin. The tuning of this parameter's value is a problem, as it has no intuitive meaning. In order to find its optimal value, it is often necessary to perform an exhaustive search in the parameter space, using a cross validation for instance (see below).

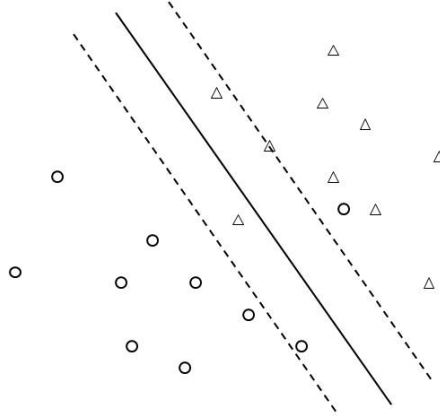


Figure 2.3: An example of non-linearly separable problem, with the introduction of slack variables.

But even with the soft margin technique, it is still not possible to solve every classification problem. For instance, if one class of the data lies in a circle, and the other outside, it is not possible to find a hyperplane that classifies correctly the objects. One idea is to map the objects in another space, called the feature space, which is often higher-dimensional, with the help of a non-linear mapping ϕ . The objective is to find a feature space in which the data are linearly separable. We saw previously that the dual formulation of the optimization problem only depends on the dot product of the data $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. If those data are mapped to the feature space, this product becomes $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

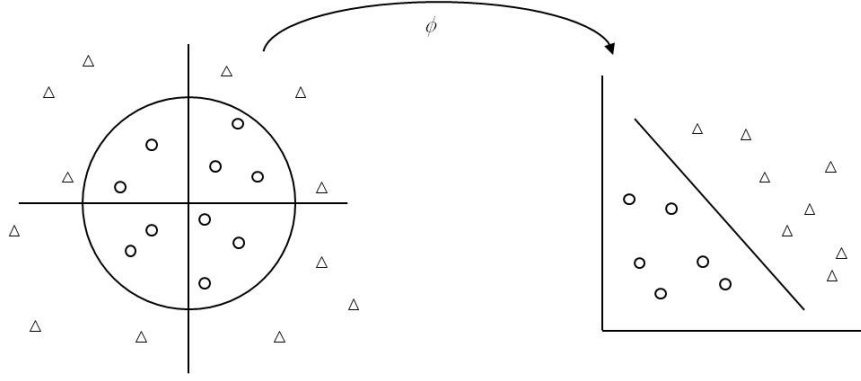


Figure 2.4: An example of non-linearly separable problem, illustrating the kernel trick. On the left are the original data in the input space, on the right are the data mapped into feature space, where they become linearly separable.

If a function k , called kernel function, is defined by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle, \quad (2.14)$$

the decision function becomes

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^m y_i \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle + b \right) = \text{sgn} \left(\sum_{i=1}^m y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) + b \right). \quad (2.15)$$

The optimization problem is now

$$\underset{\alpha \in \mathbb{R}^m}{\text{maximize}} \quad \mathbf{W}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (2.16)$$

$$\text{subject to } \alpha_i \geq 0 \text{ for all } i = 1, 2, \dots, m \text{ and } \sum_{i=1}^m \alpha_i y_i = 0. \quad (2.17)$$

We can see in this classification method that the objects to classify are mapped to a higher-dimensional feature space, using a non-linear map ϕ . But it is not necessary to know explicitly ϕ , because only the kernel function is needed. This is called the *Kernel Trick*. The kernel function has to meet certain conditions to represent a dot product in a higher-dimensional space. It must be symmetric :

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i) \quad (2.18)$$

and positive semidefinite

$$\sum_{i=1}^m \sum_{j=1}^m c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \quad (2.19)$$

where $c_i, c_j \in \mathcal{R}$. In fact, when using a SVM, the key task is to choose the kernel adapted to the situation. Some of them are widespread in the machine learning community :

- the polynomial kernel: $k = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$. If $c = 0$ and $d = 1$, this kernel is known as the linear kernel,

- the Gaussian kernel: $k = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$.

In the next section, we review some of the most well-known graph kernels, which can be used in conjunction with SVMs for graph classification (see Section 2.4).

2.2.5.2 Review of graph kernels

The first kernel defined on structured objects is the R-convolution kernel [62], and is a basis for several other kernels that are detailed hereunder. Indeed, it computes the similarity of two objects by using the similarities between the compounds resulting of all the possible decompositions of these objects. It therefore computes the similarities between subgraphs belonging to two graphs, whatever these subgraphs are (trees, walks, paths,...). But, as this all subgraphs kernel is equivalent in complexity to decide if two graphs are isomorphic, usually new kernels are derived from the all subgraphs kernel by restricting themselves to only specific types of subgraphs which are manageable from a computation time point of view.

The first kernels based on walks, which count the number of similar random walks (for instance with similar labels on nodes/edges) in two graphs to determine their global similarity, are the ones from Gartner [55], which uses the k -th power of the adjacency matrix to count k -steps walks, and Kashima [77] which considers random walks as sequences of labels (see Section 2.2.5.3 for details). These kernels have a time complexity of $\mathcal{O}(n^6)$. [160] brought down the complexity of these kernels to $\mathcal{O}(n^3)$ using Kronecker products. Those kernels also suffer of two defaults: halting and tottering. Halting has its origin in the fact that the number of walks of length superior to 1 is infinite, if cycles are allowed. A decaying factor is then usually used to favor shorter walks, which become almost the only kind of walks taken into account by the kernel. Tottering has another origin: as the repetition of nodes or edges is allowed, a random walker can stay “stuck” in a loop, counting an element artificially a great number of times. This may increase artificially the kernel value between two graphs. Variants of these kernels have appeared over time, trying to resolve the limitations cited above. [60] used random walks of fixed length in an image classification task, and [103] developed an extension of marginalized graph kernel which prevent tottering, using the Morgan index (see Section 2.2.3.2) to relabel the nodes. Another kernel using walks and paths is the shortest-path kernel [15], which transforms original graphs into “shortest-path graphs”, i.e., completely connected graphs where all edges bear the shortest-path distance between the two nodes it connects, which are then compared across the two graphs (see Section 2.2.5.4 for more details). The complexity of this algorithm is $\mathcal{O}(n^4)$. Other kernels have been created, specialized in the domain they study, as [123] which uses molecular descriptors and counts labelled paths of fixed length.

Another kind of graph kernel uses cycles in graphs to determine a similarity measure [66]. It counts pairs of similar cycles in two graphs, but is unfortunately *NP-hard*. [56] developed a linear time kernel based on cycles and applied to chemoinformatics, using molecular information to increase the discriminant power. A kernel counting graphlets, i.e., all small subgraphs of size $k \in \{3, 4, 5\}$ was introduced by Shervashidze and Borgwardt [148], and has a time complexity of $\mathcal{O}(n^{2k})$.

The last class of kernels reviewed in this section considers subtree patterns. The first kernel using that kind of patterns was the one from Ramon [124]. It counts, for all couple of nodes v from G_1 and v' from G_2 , all pairs of matching structures in subtrees rooted at v and v' . Again, this kernel has a quite high time complexity: $\mathcal{O}(N^2 n^2 h 4^d)$, where N is the number of graphs, n the number of nodes, h is the maximal height of the subtrees in the graph, and d is the maximal degree in the whole dataset. [104] refined this kernel by imposing α -ary subtrees (i.e., subtrees with at most α children per node). The Weisfeiler Lehman kernel [147] also uses subtrees, and combines labels on vertices and on their neighborhoods repeatedly, in order to obtain sequences of labels representing each a substructure in the graphs. Two sequences are then compared between graphs to obtain a similarity measure. This kernel is detailed hereunder.

We now detail three classical graph kernels: the marginalized graph kernel, the shortest-path kernel and the Weisfeiler-Lehmann kernel. These three methods serve either as inspiration for the new kernels that are described in the next section, or as baselines for the experiments.

2.2.5.3 Marginalized kernel

The main idea behind this kernel [77] is to perform random walks on labelled graphs (where both nodes and edges are labelled). While a random walker goes through a graph, he produces sequences of labels as he traverses nodes and edges, like the one hereunder:

$$(A, b, C, a, D, e, C),$$

(A, C, D) being node labels, and (a, b, e) being edge labels. This algorithm extracts all the label sequences from a graph (see Figure 2.5), which can be numerous, especially when the size of the graph increases. The length of these walks can also be infinite, if there are loops in the graph. We can already foresee a major drawback of this kernel, namely the scalability. All those sequences from a graph G_1 are then compared to the ones of a graph G_2 . More precisely, the node label n_{1,G_1} from G_1 is compared to node label n_{1,G_2} of G_2 , then edge label e_{1,G_1} is compared to e_{1,G_2} , etc. A Dirac kernel is used to evaluate the similarity of those elements, so the result is one if the labels are the same (on nodes or edges), and zero if the labels are different:

$$K = \delta(n_{1,G_1}, n_{1,G_2})$$

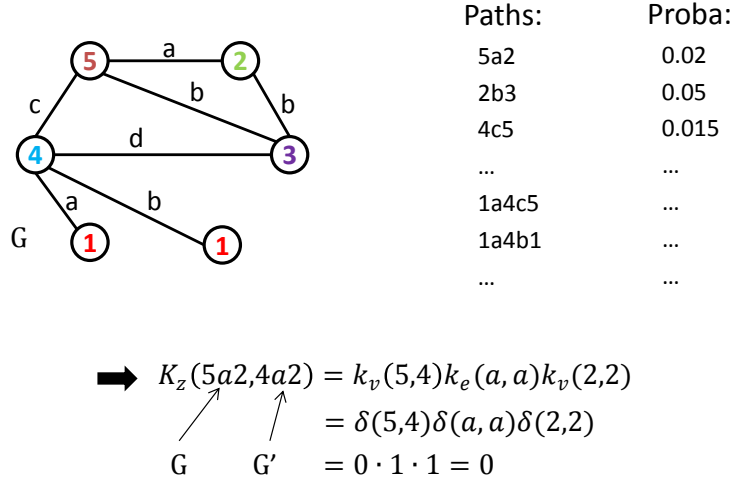


Figure 2.5: Extraction of the labelled sequences from graph G , and computation of the similarity between two sequences from graphs G and G' .

Figure 2.5 applies this to two sequences $(5, a, 2)$ and $(4, a, 2)$.

The label sequences are also weighted by their probability of appearance. Indeed, if a random walk is performed on a graph, each node in that graph has a probability p_s that this walk starts at this particular node, a transition probability p_t of going from that node to a neighbor node, and a termination probability p_q , which represents the probability that the walk ends at this node. If there is no prior information about these probabilities, p_s can be set to the uniform distribution, p_t is the inverse of the degree of the initial node (before transition), and p_q is a parameter that can be tuned, to favor short or long walks. The weighted similarities between two walks are then summed over all sequences of the two graphs, and this gives us a global similarity score:

$$\begin{aligned} K(G, G') &= \sum_{l=1}^{\infty} \sum_{w^{G_1}} \sum_{w^{G_2}} p_s(w_1^{G_1}) \prod_{i=1}^l p_t(w_i^{G_1} | w_{i-1}^{G_1}) p_q(w_l^{G_1}) \\ &\quad \cdot p_s(w_1^{G_2}) \prod_{i=1}^l p_t(w_i^{G_2} | w_{i-1}^{G_2}) p_q(w_l^{G_2}) \\ &\quad \cdot K(n_{w_1^{G_1}}, n'_{w_1^{G_2}}) \prod_{k=2}^l K(e_{w_{k-1}^{G_1}}, e'_{w_{k-1}^{G_2}}) K(n_{w_k^{G_1}}, n'_{w_k^{G_2}}), \end{aligned}$$

where w^{G_1} is a walk in G_1 , w^{G_2} is a walk in G_2 , w_i is the i th vertex in the sequence, and n, e are the labels on the nodes and edges. The enumeration of all the label sequences is unfortunately very computationally expensive, and the pair comparison between all the sequences from two graphs makes the time to compute this kernel grows strongly. Indeed, the complexity of this kernel is $\mathcal{O}(n^6)$, where n is the number of nodes (if G_1 and G_2 have the same number of nodes). This makes this kernel not tractable with databases containing more than a few hundred molecules.

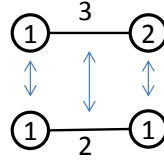
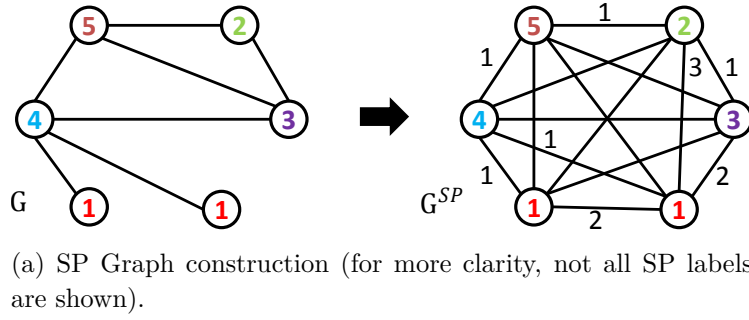


Figure 2.6: Construction of the Shortest-Path Graph from initial graph G (a) and comparison of 1-Walks (b).

As already stated, a modification of the marginalized kernel [160], brought the complexity down to $\mathcal{O}(n^3)$, thanks to the use of direct graph product, and the resolution of a linear equations system.

2.2.5.4 Shortest-Path kernel

The first step of this kernel consists in transforming an original labelled graph G into a “Shortest-Path (SP) graph” G^{SP} [15] (see Figure 2.6(a)). This new graph has the same set of vertices (or nodes) than the original one. The set of edges is redefined: there is an edge between two vertices if there was a path connecting them in the original graph. The labels on the new edges are the shortest-path distances between the nodes in G . If the edges of the original graphs do not bear any label which correspond to a distance, the shortest path between two nodes can simply be considered as the minimum number of edges between these two nodes. The all-pairs shortest paths can be computed using a Floyd-Warshall algorithm, which has a complexity of $\mathcal{O}(n^3)$.

Then each 1-walk (made of an edge, the two corresponding vertices, and their labels) of a graph G_1^{SP} is compared to those of graph G_2^{SP} (see Figure 2.6(b)), using, for example, a Dirac kernel, as in the marginalized kernel. As the shortest-path distances are real

numbers, a Gaussian (RBF) kernel can be used to compare them:

$$k = \exp \left(-\frac{\|d_i^{G_1} - d_j^{G_2}\|^2}{2\sigma^2} \right),$$

where $d_i^{G_1}$ is a SP distance in graph G_1 and $d_j^{G_2}$ is a SP distance in graph G_2 . The comparison between two walks, which provides a similarity measure, is the product between the node kernels and the edge kernel. It will give a zero result if one (or both) of the pairwise node labels are different, or if the distances are the same. The sum of the comparison on all the 1-walks is used as a similarity measure between these two graphs. The comparison of all the SP walks can be performed in $\mathcal{O}(n^4)$, which is global complexity of the SP kernel. Indeed, after the transformation to shortest-path graphs, the latter are completely connected. And if both graphs have n nodes each, then the pairwise comparison of all the SP distances can only be done in n^4 operations.

2.2.5.5 Weisfeiler-Lehman kernel

The Weisfeiler-Lehman kernel, developed in [147], is inspired by a test of isomorphism between two labelled graphs, the Weisfeiler-Lehman test (unlabelled graphs can also be handled, labels being initialized to node degrees). The key idea of this algorithm is to concatenate the label of a node with the ones of its neighbors, to obtain a label representing the local structure of the graph around that node. The first step of this test is to list all types of nodes present in the graph. and to associate them to an integer, via a hash table for instance. Each new label is then aggregated with its nearest neighbors, to form a new list of labels, representing the local neighborhood of the nodes. At each iteration, the size of the neighborhood increases, taking into account wider and wider subgraph structures. When the respective lists of labels of the two graphs differ, then the test is able to say that the two graphs are not isomorphic. If the set of labels is still the same after n iterations, the algorithm cannot tell if the graphs are isomorphic or not.

This isomorphism test was transformed to a graph kernel simply by counting the number of common transformed labels between two graphs. Each graph is represented as a vector, in which each element represent the number of occurrence of a newly created label type. This vector is updated at each iteration of the algorithm, which goes a step forward to the next “ring” of neighbors (see Figure 2.7). In the end, one obtains a matrix M in which each row represent a graph, each column a type of “concatenated” label, and each element of M is the number of times a label occurs in a certain graph. The global measure of similarity between two graphs can be obtained by using a linear kernel, which amounts to multiply the matrix M with its transposed matrix.

This kernel’s runtime complexity is of $\mathcal{O}(hm)$, where h is the number of iterations, i.e., the size of the neighborhood, and m is the number of edges. It is one of the most accurate

and fastest kernel known in the literature, and it is used for comparison throughout the different experiments we performed.

2.3 Proposed kernels

This section is devoted to the presentation of three new graph kernels we defined in order to compare graphs. In order to compare the performance of the various kernels, real life datasets, which are referenced by many authors in the literature, are used in Section 2.4.

2.3.1 Modified Weisfeiler-Lehmann kernel

As a reminder, the original Weisfeiler-Lehman kernel [147] computes the global similarity between two graphs simply by performing a dot product between the vectors representing the graphs (linear kernel). This is quite efficient, as all the vector products can be done in one pass, multiplying the global matrix with its transposed matrix. But one can wonder if the comparison between the label vectors can be improved. One idea to refine this process is to consider the vectors as frequency distribution of labels, per type of label. Each element of those vectors would represent the frequency of occurrence of a label in a graph. It then becomes possible to compare these distribution with well-known tools, as the Kullback-Leibler divergence [82], or its symmetrized version, the Jensen-Shannon divergence [92]:

$$JS(p_1, p_2) = H\left(\frac{p_1 + p_2}{2}\right) - \frac{H(p_1) + H(p_2)}{2}, \quad (2.20)$$

where p_1 and p_2 are two probability measures, and H is the Shannon entropy, defined by:

$$H(X) = -\sum_{i=1}^n P(X = x_i) \log(P(X = x_i)) = -\sum_{i=1}^n p_i \log(p_i). \quad (2.21)$$

For a discrete random variable X , comorting i symbols, p_i is the probability of observing object x_i , here a particular label. We still have to transform that divergence into a similarity measure, or a kernel. We use the same method as the one shown in [108]:

$$K^{JS}(p_1, p_2) = \ln(2) - JS(p_1, p_2). \quad (2.22)$$

We modified the original WL kernel in that direction, computing the Jensen-Shannon divergence for all pairs of graphs, and converting the results into a kernel.

2.3.2 Randomized Shortest-Path kernel

The first idea that led to this kernel is similar to the one used in the shortest-path kernel [15]. In that paper, the initial graphs are transformed to shortest-path graphs, using the

Floyd-Warshall algorithm, for instance. Our idea consists in the use of another distance between nodes of a graph than the shortest-path one, by introducing some entropy into the computation of those distances. This idea was inspired by a paper of Yen et al. [174], in which a new dissimilarity between the nodes of a graph is introduced, generalizing the shortest-path and the commute-time distances. This algorithm lets, for instance, a traveller explore the graph and discovers new “roads”, instead of going straight to another node with a shortest-path distance. This concept is also helpful in domains where artificial intelligence is needed, as chess game. In order to avoid too straightforward strategies from the computer, one can introduce some entropy, or disorder, to create more unpredictable decisions.

Our aim is therefore to adapt the dissimilarity measure presented above (applied for ongoing pairs of nodes), to obtain a distance measure between pairs of graphs, by using the same methodology as Borgwardt. For each dataset, we therefore transform original graphs into “randomized shortest-path” (RSP) ones, computing the all-pairs RSP distances on these graphs. The “amount” of entropy injected in the calculation is controlled by the parameter θ , according to the desired degree of randomness. When $\theta = 0$, the distances reduce to the commute-time distances, and when θ is large, they tend to shortest-path distances. The resulting graphs are completely connected, since every node is connected to every other by an edge labelled with the RSP distance. Thus, a list of distances is obtained for each graph, which has to be compared in order to obtain the global similarity between two graphs. In [15], this is done by comparing “1-walks” (see Section 2.2.5.3) between them.

Since quite computationally demanding, computing the comparison between all the randomized shortest paths between two graphs has to be avoided. The distances are therefore discretized in order to produce frequency distributions of distances. At first, all the edge labels existing in a dataset, per type on nodes labels, are collected. Then, based on the percentiles, the boundaries of each distance class are determined, and applied to each RSP graphs. Distributions of frequencies concerning the distances, one for each type of node label, are thus obtained. These distributions are weighted by the a-priori probabilities to find a type of node label in the whole dataset. Finally, the Jensen-Shannon kernel between two graphs is computed.

2.3.3 Graph Features kernel

As real life graphs can count several thousands of nodes (and more), it is essential that a kernel present a good scalability. In this vein, the Graph Features (GF) Kernel assumes that a single, fast to compute, global characteristic of the graphs, could lead to a measure of similarity. The first characteristic to be checked is the Frobenius norm of the graphs’ adjacency matrices, which are compared using a RBF kernel, and the classification is

done via a SVM (LibSVM on Matlab). We suppose that similar graphs should have similar norms. Despite its simplicity, the preliminary results are surprisingly good on some datasets, very close to those obtained by state-of-the-art methods. Moreover, the speed is clearly one of the strengths of this kernel, as it can be computed in linear time of the number of nodes. Note that this aspect could still be improved, since using a RBF kernel implies the comparison of all the pairs of feature vectors. Instead, using a linear kernel permits to compute all the similarities in one pass, using just a matrix product. Another advantage of the latter kernel is the absence of parameter to tune, unlike the RBF kernel, which depends on the σ parameter. We then tried to obtain higher accuracy by adding other features characteristics of the graphs, as the number of nodes, edges, the density of the graphs, the degree distribution,... Only characteristics that could be computed in linear time were chosen and the feature vector used in the experiments is composed of:

- the Frobenius norm (NF) (of a matrix $\mathbf{M}$):

$$\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |m_{i,j}|^2},$$

- the Manhattan norm (NM):

$$\|\mathbf{M}\|_1 = \sum_{i=1}^n \sum_{j=1}^n |m_{i,j}|,$$

- the Infinity norm (NI):

$$\|\mathbf{M}\|_\infty = \sqrt[n]{\sum_{i=1}^n \sum_{j=1}^n |m_{i,j}|^n} = \max_i \sum_{j=1}^n |m_{ij}|,$$

- the 2-norm (N2), which is the magnitude of the matrix' largest singular value.
(Those four norms are applied to the adjacency and Laplacian matrices of a graph G)
- the number of nodes in the graph (Ni),
- the number of edges in the graph (Ei),
- the distribution of the degrees of the nodes,
- the distribution of the labels of the nodes,
- the density of the graph (number of edges of the graph divided by the number of edges of a complete graph with the same number of nodes) (Dens),

- the minimum and maximum degree of the nodes in the graph (MinDeg and MaxDeg).

The final kernel between two graphs is then computed using, for instance, a linear kernel between the two feature vectors corresponding to the two graphs, which is equivalent to a dot product between the two feature vectors. As the optimal set of characteristics can be different from one dataset to another, we developed a simple feature selection algorithm. This algorithm computes the correlation, with a built-in Matlab tool, of a feature vector with the class label vector, for all the graph features. Then all the features that present a correlation coefficient that is above a certain threshold are selected to participate to the elaboration of the kernel matrix. The threshold is determined via cross-validation. Moreover, as all the characteristics do not have the same order of magnitude, each of them is normalized separately, by removing the mean and dividing by the standard-deviation of each feature.

We also retrieved the features which are the most discriminant when performing the classification, using the LibSVM toolbox. The five most important features for each dataset are shown in the Figure 2.8.

	1st	2nd	3rd	4th	5th
MUTAG	Ni	N2A	Ei	NFL	N2L
PTC	Dens	N2A	NFA	NFL	Ei
NCI1	MinDeg	Ei	NIA	NMA	MaxDeg
D&D	NFA	N2A	NFL	N2L	Dens

Figure 2.8: Ranking of the features based on their importance in the classification, on four datasets (NxL stands for Norm x of adjacency/Laplacian matrix).

The results show that the set of important features is different for each dataset, but that the Frobenius norm and the 2-norm (both for adjacency and Laplacian matrices) are the most frequent (except for NCI1 where they do not appear in the top 5). This may indicate that a very simple measure as the Frobenius norm, which is the sum of the squared elements of the matrix, suffices to produce a correct classification in a very fast way. The Frobenius norm is also the square root of the trace of a matrix $\mathbf{M}$ multiplied by its transpose

$$\|\mathbf{M}\|_F = \sqrt{\text{Tr}(\mathbf{M}\mathbf{M}^T)},$$

where the trace of a matrix is the sum of its eigenvalues. This, together with the good results of the 2-norm, brings us back to spectral analysis of matrices, which is a domain that could be further studied in order to produce interesting features for graph classification.

The distribution of labels and degrees were not included in the feature ranking procedure, as they represent by themselves several features (each one for a type of labels, or class of degree), and retrieving one label or one value of degree would not make sense.

2.4 Experiments

This chapter describes the datasets used to assess the quality of the kernels developed in the previous section, as well as the procedure applied for this evaluation.

2.4.1 Datasets

Four datasets are used during the experiments. They all contain molecular graphs stored as adjacency matrices, adjacency lists and node labels lists, and a binary vector encoding the activity of a molecule concerning a particular task. The first one is the MUTAG dataset [32]. It contains 188 molecular graphs related to molecules that can present a mutagenic activity on bacterium *Salmonella Typhimurium*. The second one is the PTC MR dataset [63], which contains 344 molecules resulting from pharmaceutical experiments on anticancer activity. The original dataset included experiments on male and female rats and mice, but we only used the male rat (MR) part here. This dataset is known to be hard to classify. The third dataset is the NCI1 [163] which contains about four thousands graphs also representing compounds possessing an anticancer activity or not. And at last, the D&D (Dobson & Doig) [38] dataset is composed of about one thousand protein structures. Here, the nodes of the molecular graphs represent amino acids. There is a link between two nodes if they are sufficiently close (less than 6 Ångstroms). We will try here to predict if a protein is an enzyme or not. Below is a recapitulative table including the most important information about the datasets:

Name	MUTAG	PTC	NCI1	D&D
Number of graphs	188	344	4110	1178
Max number of nodes	28	64	111	5748
Average number of nodes	18	14	30	284

Figure 2.9: Characteristics of datasets.

2.4.2 Experimental setup

We assessed the behavior of our kernels on these four datasets with a binary classification task.

Each dataset includes a target vector indicating if a compound present a desired activity or not. We performed the classification using a support vector machine (SVM) using the Matlab implementation of LIBSVM. These results were compared with some classical graph kernels found in the literature, representing diverse families: the marginalized kernel (based on walks), the shortest-path kernel (based on paths), and the Weisfeiler-Lehman kernel (based on subtrees). Double cross-validation is used: the datasets are divided in ten folds, one remains for the purpose of test (called the test set) while the other nine folds, forming the training set, are used for training. The training set is again divided into ten folds, and cross-validation is performed in order to identify the combination of parameters giving the best accuracy results. Those parameters, i.e., C from the SVM, σ from the RBF kernel, ..., are then used on the external test set. This double cross-validation process is repeated ten times, in order to reduce the variability of the results, due to the small sizes of the datasets, in particular MUTAG and PTC.

2.4.3 Results

The results shown hereunder are the accuracy (percentage of good classification), and total computation time, obtained when applying double cross validation, except for the * cases, where a double CV was not performed, because of the slowness of the kernel, and the large size of the datasets. In the case of the marginalized kernel, no tests could be done on the NCI1 and D&D datasets, as this kernel took more than 2 days to run the experiments, and we decided to stop it. The best results per dataset are displayed in bold in Figure 2.10.

Dataset	MUTAG	PTC	NCI1	D&D
WL Mod Kernel	88.19	59.36	86.03	80.31
RSP Kernel	87.09	56.23	63.78*	77.75*
GF Kernel	89.69	60.72	68.43	79.10
WL Kernel	88.20	64.18	86.04	79.78
Marginalized Kernel	87.84	60.14	/	/

Figure 2.10: Classification accuracy of the different kernels (in percents).

Figure 2.10 shows that the results obtained by the modified WL kernel (WL Mod

Kernel) are not substantially better than the original WL kernel, and the RSP kernel displays the worse results. The GF kernel provides the best results on the MUTAG dataset, is competitive on the D&D and PTC datasets but behaves not so well on the NCI1 dataset. The simple features extracted from the graphs may not have sufficient discriminant power to classify them.

Concerning the computation time (Figure 2.11), GF kernel is way fastest than the other kernels, even compared to the Weisfeiler-Lehman, which is one of the fastest graph kernel in the literature. This is a consequence of the simplicity of the features used in our kernel. The computation time when using the D&D dataset (resp. the NCI1 dataset) is about 9 seconds (resp. 6 seconds), where the second fastest kernel, the original WL kernel, takes more than 11 minutes (resp. 7 minutes). Concerning small datasets, the GF kernel takes only a fraction of second to realize the computation. WL Mod kernel is slower compared to the original WL one, because it does not use a linear kernel that computes the kernel matrix in one pass. Instead, it compares graphs pairwise, and is slower as the number of graphs increases. RSP kernel, at last, is plagued by the computation of the RSP distances, which increases as the cube of the size of the graph.

Dataset	MUTAG	PTC	NCI1	D&D
WL Mod Kernel	24s	64s	2h04min	25min
RSP Kernel	10s	21s	1h16min	17h06min
GF Kernel	0.1s	0.2s	6s	9s
WL Kernel	6s	10s	7min	11min
Marginalized Kernel	50s	5min56s	> 2days	> 2days

Figure 2.11: Computation time of the different kernels.

It should be noted that, while we were developing the GF kernel, we saw that Li et al. [89] published a paper using almost the same idea, except that they use more complex features extracted from the graphs. Their results are thus better (accuracy), but the computational time is also slower, than our GF Kernel.

In order to determine to which extent the results given by the different kernels are related to each other, the Spearman's correlation was computed between the class vectors of each method (other correlations like Kendall, Linear,... give the same trends) on two datasets: PTC and NCI1 (see Figure 2.12 and 2.13). We can observe that the different methods are not much correlated, except for the WL kernel and the Mod WL kernel, which is quite normal as they are based on the same method.

	RSP Kernel	GF Kernel	WL Kernel	WL Mod Kernel
RSP Kernel	1.0000	-0.0714	0.0986	0.1330
GF Kernel	-0.0714	1.0000	0.1707	-0.0754
WL Kernel	0.0986	0.1707	1.0000	0.4890
WL Mod Kernel	0.1330	-0.0754	0.4890	1.0000

Figure 2.12: Correlation between the different kernels, on PTC dataset.

	RSP Kernel	GF Kernel	WL Kernel	WL Mod Kernel
RSP Kernel	1.0000	0.0354	-0.1010	-0.0912
GF Kernel	0.0354	1.0000	0.3037	0.2783
WL Kernel	-0.1010	0.3037	1.0000	0.7767
WL Mod Kernel	-0.0912	0.2783	0.7767	1.0000

Figure 2.13: Correlation between the different kernels, on NCI1 dataset.

2.5 Chapter conclusion

In this chapter, we developed three new graph kernels in order to be able to compare graphs. The first one is based on the Weisfeiler-Lehman kernel [147], refining it by using the distribution of the structure labels, and comparing them with the Jensen-Shannon divergence. The second one is based on the Shortest-Path kernel [15], where we replaced the shortest-path distances with the randomized shortest-path distances [174], allowing some exploration on the graph. At last, the third kernel developed uses graph features which can be computed in linear time to construct feature vectors, which are then compared with a linear kernel.

Those new kernels were compared to classical graph kernels, like the marginalized kernel, the shortest-path kernel and the Weisfeiler-Lehman kernel on a binary classification of molecules (modelled as graphs). Although the new kernels give interesting results, none of them overcame the Weisfeiler-Lehman Kernel in term of accuracy. However, the graph features kernel was designed for speed, and thus surpasses all the other kernels in test, and provides good results on three datasets on four.

Chapter 3

Bag-of-Forests similarities between nodes of a graph

Contents

3.1	Review of similarity measures on nodes of a graph	38
3.1.1	Local similarity measures	38
3.1.2	Global similarity measures	42
3.1.3	Kernel-based similarity measures	43
3.1.4	Dissimilarities interpolating between commute-time and shortest-path	46
3.2	Introduction to the Matrix-Forest Theorem	47
3.3	A new forest similarity measure	49
3.3.1	The BoF potential distance	51
3.3.2	The BoF logarithmic distance	51
3.4	Experiments	52
3.4.1	Datasets	52
3.4.2	Experimental setup	53
3.4.3	Results	54
3.5	Chapter conclusion	68

3.1 Review of similarity measures on nodes of a graph

Similarity measures between nodes of a graph are of central importance in various fields such as social networks analysis, graph clustering, web mining, etc. One possible approach to find such similarities is to use features on nodes in order to determine if two nodes are similar (or close). Another approach is to use the structure of the graphs: the more two nodes share common (sub)structures, the more similar they are. The most well-known and intuitive similarity measure using graph substructure may be derived from the shortest-path (SP) distance. Indeed, two nodes are similar if the SP distance between them is small. This distance has already been applied in Chapter 2 on global similarity measures between graphs. The similarity measures treated in the rest of this section are classified into two categories: (1) local, which uses purely local (i.e., around the nodes of interest) graph structure and (2) global which involves the whole graph structure (see [50], Chapter 3 for more details).

3.1.1 Local similarity measures

Similarity measures introduced in this section deal with undirected and unweighted graphs, except for some cases where the weighted and/or directed case is also treated. These similarity measures are based on the direct connections of the nodes on which the measure is computed. This set of directly connected nodes is called a “neighborhood”, and for node i , is denoted $\mathcal{N}(i)$. Several neighborhoods can be defined, as illustrated in figure 3.1. Considering two nodes i and k , V is the entire set of nodes constituting the graph, P is the neighborhood of node i only, while R is the one from node k only. Q is the common neighborhood of nodes i and k , while S is composed of nodes belonging to neither to P, Q or R . Several variables can be extracted from those neighborhoods. Let us call them p, q, r, s and define them respectively as the cardinality of the different sets P, Q, R, S , i.e., $p = |P|$, etc. These variables can also be written in terms of the elements of the adjacency matrix a_{ij} (containing only 0,1 values), as seen in Figure 3.2, for nodes i and k .

Let us now review the most well-known similarity measures based on the neighborhoods of nodes.

3.1.1.1 Common Neighbors

This similarity measure is one of the simplest, as it only count the number of common neighbors p between two nodes i and k :

$$\text{sim}(i, k) = p. \quad (3.1)$$

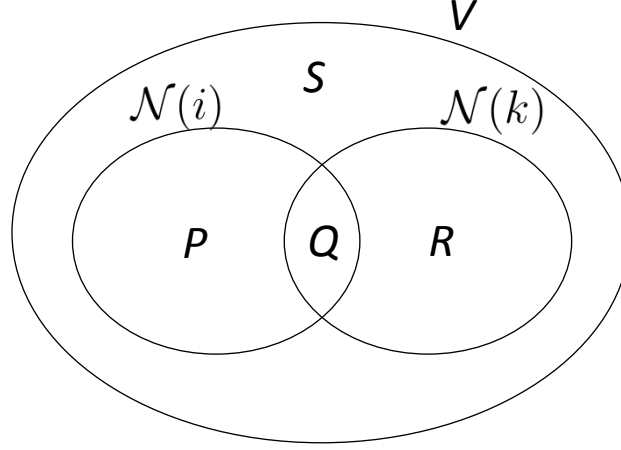


Figure 3.1: Diagram illustrating V , the set of nodes on the graph, and the neighborhoods of nodes i and k .

$$\left\{ \begin{array}{l} p = \sum_{j=1}^n a_{ij}a_{kj} = \sum_{j=1}^n a_{ij}a_{jk} \\ q = a_{i\bullet} - p \text{ where } a_{i\bullet} = \sum_{j=1}^n a_{ij} \\ r = a_{\bullet k} - p \text{ where } a_{\bullet k} = \sum_{j=1}^n a_{jk} \\ s = n - (a_{i\bullet} + a_{\bullet k} - p) \\ p + q = a_{i\bullet} \\ r + s = n - a_{i\bullet} \\ p + r = a_{\bullet k} \\ q + s = n - a_{\bullet k} \\ n = p + q + r + s \end{array} \right.$$

Figure 3.2: Different neighborhood indicators defined as adjacency matrix elements of the corresponding graph.

In the case of weighted graphs, this similarity measure can be written using vectors $\mathbf{v}_i$ and $\mathbf{v}_k$, designating rows i and k of the adjacency matrix (where the non-zero elements are the weights beared by the edges). The common neighbors similarity can thus be simply rewritten

$$\text{sim}(i, k) = \mathbf{v}_i^T \mathbf{v}_k. \quad (3.2)$$

[115] used this similarity measure to study the time evolution of scientific collaboration networks in physics and biology by analyzing the common neighbors of scientists, and showed that the more scientists have collaborators in common, the more the probability they collaborate increases.

Normalized versions of this similarity measure exists in the literature, dividing for instance the number of common neighbors p by the total number of nodes in the graph, n [74]. Other normalized measures using the common neighborhoods are detailed hereunder.

3.1.1.2 Cosine coefficient

This coefficient normalizes the number of common neighbors p between nodes i and k by the product of the square root of the total number of neighbors of i , which is $p + q$, by the square root of the total number of neighbors of k , which is $p + r$:

$$\text{sim}(i, k) = \frac{p}{\sqrt{p+q}\sqrt{p+r}}. \quad (3.3)$$

In the case of weighted graphs, this can be written

$$\text{sim}(i, k) = \frac{\mathbf{v}_i^T \mathbf{v}_k}{(\|\mathbf{v}_i\| \|\mathbf{v}_k\|)}, \quad (3.4)$$

which measures the strength and the direction of a linear relationship between two variables. Here, as the nodes i and k can be represented as vectors in the node space, the cosine coefficient represents the angle between these two vectors, and the smaller the angle is, the more similar the two nodes are.

3.1.1.3 Jaccard index

The Jaccard index (see [70]) divides the number of common neighbors between i and k by the number of nodes belonging to the neighborhood of i , k and both i and k :

$$\text{sim}(i, k) = \frac{p}{p + q + r}, \quad (3.5)$$

or in vectorial form,

$$\text{sim}(i, k) = \frac{\mathbf{v}_i^T \mathbf{v}_k}{(\|\mathbf{v}_i\|^2 + \|\mathbf{v}_k\|^2 - (\mathbf{v}_i^T \mathbf{v}_k))}. \quad (3.6)$$

3.1.1.4 Dice coefficient

Also called Sorensen index, the Dice coefficient (see [34]) differs from the Jaccard index only by multiplying the number of common neighbors p by two:

$$\text{sim}(i, k) = \frac{2p}{2p + q + r}. \quad (3.7)$$

3.1.1.5 Tversky index

This similarity measure [158] uses two parameters $\alpha, \beta \geq 0$ in order to independently adjust the weights of the neighborhood of nodes i and k , and thus can be seen as generalized version of the Jaccard index and Dice coefficient :

$$\text{sim}(i, k) = \frac{2p}{p + \alpha q + \beta r}, \quad (3.8)$$

as when $\alpha = \beta = 1$ the Tversky index is equivalent to the Jaccard index, and when $\alpha = \beta = 0.5$ it is equivalent to the Dice coefficient.

3.1.1.6 Hub Promoted and Hub Depressed indices

The hub promoted index [127] was first proposed to evaluate the topological overlap of pairs of substrates in metabolic networks (hubs are defined as nodes having many connections, or equivalently having a high degree). This index assigns higher similarity scores between nodes and hubs since the denominator depends on the minimum degree of those two nodes:

$$\text{sim}(i, k) = \frac{p}{\min(p + q, p + r)}. \quad (3.9)$$

In an opposite way, the hub depressed index [95] will assign a lower score for the similarity between two nodes if one of the nodes is a hub, as the denominator is the maximal degree between the two nodes:

$$\text{sim}(i, k) = \frac{p}{\max(p + q, p + r)}. \quad (3.10)$$

3.1.1.7 Preferential attachment index

This index, defined in [179] and coming from the scale-free networks field, corresponds to the product of the degrees of the two nodes:

$$\text{sim}(i, k) = (p + q)(p + r). \quad (3.11)$$

3.1.2 Global similarity measures

In this section, all nodes present in a (undirected) graph are involved to create a similarity measure between two specific nodes.

3.1.2.1 Katz index

The Katz index [79], was first proposed by Katz in 1953 in the social-science field as a centrality measure. This index can also be seen as a way to measure similarities, counting weighted paths of all lengths between two nodes. The Katz similarity matrix can be defined by

$$\mathbf{K} = \alpha \mathbf{A} + \alpha^2 \mathbf{A}^2 + \alpha^3 \mathbf{A}^3 + \cdots + \alpha^t \mathbf{A}^t + \cdots = (\mathbf{I} - \alpha \mathbf{A})^{-1} - \mathbf{I}, \quad (3.12)$$

where α is an attenuation factor, giving less importance to longer paths, and $\mathbf{A}$ is the adjacency matrix. It is a well-known result that i, j element of matrix $\mathbf{A}^t$ designates the number of paths of length t between nodes i and j , and the Katz matrix is simply the sum of all paths of length one to infinity. Moreover, for this series to converge, α must be smaller than the spectral radius (the inverse of the greatest eigenvalue) of $\mathbf{A}$. A variant of the Katz index (see [88]) does not take into account all paths, but sets the maximal length of the paths to a certain threshold, taking into account more local information:

$$\mathbf{K} = \alpha^2 \mathbf{A}^2 + \alpha^3 \mathbf{A}^3, \quad (3.13)$$

here with paths of length two and three.

3.1.2.2 The commute-time distance

Let us denote by $m(i, j)$ the hitting time, which is the number of steps a random walker has to perform in average to go from node i to arrive in node j for the first time (see for instance [17] or [118]). The average commute time is the symmetrized version of the hitting time :

$$n(i, j) = m(i, j) + m(j, i). \quad (3.14)$$

This is thus the average number of steps for a random walker to go to node j and back, from node i . It can be shown that the average commute time is a distance measure between nodes of a graph (and can therefore be transformed to a similarity measure when needed), since it respects for any nodes i, j, k the following properties:

$$\begin{cases} n(i, j) \geq 0 \\ n(i, j) = 0 \text{ iff } i = j \\ n(i, j) = n(j, i) \\ n(i, j) \leq n(i, k) + n(k, j) \end{cases}$$

However, when the number of nodes in the graph is very large, the commute-time distance converge to an expression that do not take into account the global structure of the graph [161]. Indeed, when the size of the graph grows the commute-time distance only depends on the degrees of the nodes of interest. In these cases, the commute time is inaccurate as it does not provide information about the structure of the graph, but only local structure information.

3.1.2.3 The SimRank similarity measure

The SimRank similarity [72] between two objects is defined recursively as the average similarity between their neighbors or, in a simpler way, “two objects are similar if they are related to similar objects” [72]. This recursive relation used to compute the SimRank similarities between two nodes i' and j' of a weighted directed graph can be written [50]

$$k_{i'j'}(t+1) = \begin{cases} 0 & \text{if } \mathcal{Pred}(i') = \emptyset \text{ or } \mathcal{Pred}(j') = \emptyset \\ 1 & \text{if } i' = j' \text{ and } \mathcal{Pred}(i') \neq \emptyset \neq \mathcal{Pred}(j') \\ \alpha \frac{\sum_{i \in \mathcal{Pred}(i')} \sum_{j \in \mathcal{Pred}(j')} a_{ii'} k_{ij}(t) a_{jj'}}{\sum_{i \in \mathcal{Pred}(i')} \sum_{j \in \mathcal{Pred}(j')} a_{ii'} a_{jj'}} & \text{if } i' \neq j' \text{ and } \mathcal{Pred}(i') \neq \emptyset \neq \mathcal{Pred}(j') \end{cases} \quad (3.15)$$

where $\mathcal{Pred}(i')$ is the predecessor of node i' . This can be expressed in matrix form, defining a new matrix $\mathbf{Q}$ whose elements are $q_{ii'} = a_{ii'}/a_{\bullet i'}$ when $a_{\bullet i'} \neq 0$, and $q_{ii'} = 0$ when $a_{\bullet i'} = 0$:

$$\begin{cases} \mathbf{K}(t) = \alpha \mathbf{Q}^T \mathbf{K}(t-1) \mathbf{Q} \\ \mathbf{K}(t+1) = \mathbf{K}(t) - \mathbf{Diag}(\mathbf{K}(t)) + \mathbf{I} \end{cases} \quad (3.16)$$

which is iterated until the convergence is reached. This similarity measure was extended by Blondel et al. [13] in order to define a new similarity measure between the nodes of two different graphs.

3.1.3 Kernel-based similarity measures

The kernel-based similarity measures can be considered as global similarity measures, as they consider all paths (direct and indirect) between nodes to assess their similarity. These measures behave in a nice way, as the more paths connecting two nodes and the more their “length” is small, the more these similarities increase. Six kernels on a graph are described in this section (see [48, 51] for more details): the exponential diffusion kernel ($\mathbf{K}_{ED}$), the Laplacian exponential diffusion kernel ($\mathbf{K}_{LED}$), the von Neumann diffusion kernel ($\mathbf{K}_{VND}$), the regularized Laplacian kernel ($\mathbf{K}_{RL}$), the commute-time kernel ($\mathbf{K}_{CT}$), the regularized commute-time kernel ($\mathbf{K}_{RCT}$) and the Markov diffusion kernel ($\mathbf{K}_{MD}$).

3.1.3.1 The exponential diffusion kernel

The exponential diffusion kernel $\mathbf{K}_{ED}$ is defined by

$$\mathbf{K}_{ED} = \sum_{k=0}^{\infty} \frac{\alpha^k \mathbf{A}^k}{k!} = \text{expm}(\alpha \mathbf{A}), \quad (3.17)$$

$\mathbf{A}$ being the adjacency matrix and expm the matrix exponential. As already seen, the element i, j of $\mathbf{A}^k$ can be considered as the number of k -steps paths between nodes i and j . This kernel therefore sums all paths of every length between two nodes, using $\frac{\alpha^k}{k!}$ factor to weight those paths. It then favors shorts paths between two nodes. As shown in [48], the $\mathbf{K}_{ED}$ matrix is positive semidefinite.

3.1.3.2 The Laplacian exponential diffusion kernel

This kernel is closely related to $\mathbf{K}_{ED}$, as is substitute the adjacency matrix with minus the Laplacian matrix in Equation 3.17:

$$\mathbf{K}_{LED} = \text{expm}(-\alpha \mathbf{L}). \quad (3.18)$$

The kernel matrix $\mathbf{K}_{LED}$ can be seen as a random walk, starting from a node and transitioning to a neighboring node with the probability α .

3.1.3.3 The regularized Laplacian kernel

The regularized Laplacian kernel [69, 150] provides relative accessibilities between nodes of a graph [24, 25]. It can be computed as follow:

$$\mathbf{K}_{RL} = \sum_{k=0}^{\infty} \alpha^k (-\mathbf{L})^k = (\mathbf{I} + \alpha \mathbf{L})^{-1}, \quad (3.19)$$

with $0 < \alpha < \|\mathbf{L}\|_2^{-1}$, with $\|\mathbf{L}\|_2^{-1}$ the spectral radius. This kernel is positive definite, as the Laplacian is positive semi-definite. This similarity measure has an interesting interpretation in terms of the matrix-forest theorem (see Section 3.2).

3.1.3.4 The von Neumann kernel

The von Neumann diffusion kernel [76] was introduced for computing document similarity from terms occurring in documents:

$$\mathbf{K}_{VND} = \sum_{k=0}^{\infty} \alpha^k \mathbf{A}^k = (\mathbf{I} - \alpha \mathbf{A})^{-1} \quad (3.20)$$

where again, $0 < \alpha < \|\mathbf{A}\|_2^{-1}$, with $\|\mathbf{A}\|_2^{-1}$ the spectral radius. $\mathbf{K}_{VND}$ is positive definite in that case.

3.1.3.5 The commute-time kernel

The commute-time kernel [49, 132] takes its name from the average commute-time $n(i, j)$, already defined in section 3.1.2. It can be shown that the average commute-time can be computed by

$$n(i, j) = \text{vol}(G)(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \quad (3.21)$$

where $\text{vol}(G)$ is the volume of the graph G ($\text{vol}(G) = \sum_{i,j=1}^n a_{ij}$), $\mathbf{e}_i$ is a basis vector in $\mathcal{R}^n$ representing node i , and $\mathbf{L}^+$ is the Moore-Penrose pseudoinverse of the Laplacian matrix and is positive semidefinite. Moreover, it can be shown that the elements of $\mathbf{L}^+$ are inner products between vectors representing the nodes, in the Euclidean space where these vectors are separated exactly by commute-time distances. The elements of $\mathbf{L}^+$ can therefore be considered as similarity measures between nodes, leading to the commute-time kernel:

$$\mathbf{K}_{CT} = \mathbf{L}^+ \quad (3.22)$$

without any parameter to tune.

As the Laplacian matrix is not invertible, and instead of taking the pseudoinverse of the matrix, another version of the commute-time kernel, called the **regularized commute-time kernel**, was introduced in [177] and investigated as a kernel in [48, 106], and uses a simple regularization framework:

$$\mathbf{K}_{RCT} = (\mathbf{D} - \alpha \mathbf{A})^{-1} \quad (3.23)$$

with $0 < \alpha < 1$, and $\mathbf{D}$ being the degree matrix. It can be shown that this kernel is positive definite. The element i, j of the kernel matrix can therefore be seen, in a random walk point-of-view, as the probability of visiting node j when starting in node i , knowing that the visiting rate to each node is accumulated, with a discount factor α (late visits are less important than early ones).

The **random walk with restart similarity** [119, 156] is very close to the regularized commute-time kernel, as it only multiplies the latter with the $\mathbf{D}$ matrix:

$$\mathbf{K}_{RCT} = \mathbf{D}(\mathbf{D} - \alpha \mathbf{A})^{-1}. \quad (3.24)$$

In terms of random walks, the random walker is here allowed to restart with probability $(1 - \alpha)$ from the starting node i at each step. The walker is thus constrained to stay in the neighborhood of i .

3.1.3.6 The Markov diffusion kernel

The Markov diffusion kernel [51] is based on a discrete-time diffusion Markov model, and is defined by

$$\mathbf{K}_{MD} = \mathbf{Z}(t)\mathbf{Z}^T(t) \text{ where } \mathbf{Z}(t) = \frac{1}{t} \sum_{\tau=1}^t \mathbf{P}^\tau \quad (3.25)$$

where $\mathbf{P}$ is the transition-probability matrix and t a parameter to tune.

3.1.4 Dissimilarities interpolating between commute-time and shortest-path

3.1.4.1 Logarithmically transformed forest similarities

Chebotarev defined a similarity matrix [21], which is also called the regularized Laplacian kernel, as

$$\mathbf{K}_{RL} = (\mathbf{I} + \alpha \mathbf{L})^{-1}. \quad (3.26)$$

A new matrix $\mathbf{S}$ can be defined by

$$\begin{cases} \mathbf{S} = (\alpha - 1) \log_\alpha \mathbf{K}_{RL}, & \text{when } \alpha \neq 1 \\ \mathbf{S} = \ln \mathbf{K}_{RL}, & \text{when } \alpha = 1 \end{cases} \quad (3.27)$$

where $\alpha > 0$ (α is the same in Equations 3.26 and 3.27), $\log_\alpha$ is the logarithmic function in base α and $\ln$ is the natural logarithm function. In order to obtain a squared distance metric, the following transformation is applied, giving the logarithmic regularized Laplacian distance matrix Δ_{LRL}

$$\Delta_{LRL} = \mathbf{diag}(\mathbf{S})\mathbf{e}^T + \mathbf{e} \mathbf{diag}(\mathbf{S})^T - 2\mathbf{S}. \quad (3.28)$$

This matrix contains distances between the nodes of a graph. Moreover, when $\alpha \rightarrow 0^+$, the distance converge to the shortest-path distance, and when $\alpha \rightarrow \infty$, to the commute-time distance.

3.1.4.2 The bag of paths framework

The bag of paths model (BoP) [52, 107] provides a distance measure between nodes of a weighted directed graph. It is based on the probability of picking a path with starting node i and ending node j from a bag, the probability of picking a path being defined by a Boltzmann probability distribution, favoring short paths. The bag of hitting paths is derived from the original BoP model, considering only paths with an absorbing ending node, i.e., a random walker only pass one time on the ending node. At first, matrices

Π whose elements are the probabilities to pick a (hitting) path beginning in node i and ending in node j in the bag. Then, two distances are derived from the probability matrix related to hitting paths, giving rise to surprisal distance and potential distance (see [52] for more details). The latter distance ranges from the shortest-path distance to commute-cost distance, depending on a parameter θ , the commute-cost between node i and node j being the expected cost a random walker came across for reaching node j and return, from node i .

3.1.4.3 The randomized shortest-path dissimilarity

Following the same framework than the bag-of-paths, the randomized shortest-path dissimilarity measure [174] also ranges from the shortest-path to the commute-cost. The randomized shortest-path (RSP) cost between two nodes is defined as the expected cost of all the hitting paths connecting those two nodes, which can be considered as a natural way for quantifying a distance between two nodes. The randomized shortest-path dissimilarity is then defined as the average RSP cost for going from node i to j and back.

In the next section, we will dive into the details of the matrix-forest theorem in order to develop a new similarity measure between nodes of a graph.

3.2 Introduction to the Matrix-Forest Theorem

Considering multigraphs, let us define by $\epsilon_{i,j}^p$ the weight of the p th edge between edges i and j . The weight of a subgraph H of a multigraph G is the product of the edges' weight of H and will be denoted by $\epsilon(H)$. The weight of an empty set is zero and the weight of a subgraph without edges is defined to one. Let us note by L^{ij} the cofactor of l_{ij} in $\mathbf{L}$, and the set of all spanning trees present in the multigraph G by $\mathcal{T}$.

The matrix-forest theorem [24, 25] is based on the well-known matrix-tree theorem from Kirchoff:

Theorem 3.1 (Matrix-Tree Theorem for multigraphs). *For any weighted multigraph G and for any $i, j \in V(G)$, $L^{ij} = \epsilon(\mathcal{T})$.*

In other words, this theorem states that the total weight of spanning trees in a multigraph G is equal to the absolute value of any cofactor of the Laplacian matrix, $\mathbf{L}$. If the weights associated to the edges are one, theorem 3.1 gives the number of spanning trees in G .

When dealing with multidigraphs (directed multigraphs), L^{ij} is again the cofactor of

l_{ij} in $\mathbf{L}$, $\mathcal{T}^i$ is the set of all spanning trees of G diverging from i , and the matrix-tree theorem becomes:

Theorem 3.2 (Matrix-Tree Theorem for multidigraphs). *For any weighted multidigraph G and for any $i, j \in V(G)$, $L^{ij} = \epsilon(\mathcal{T}^i)$.*

Let us define the matrix $\mathbf{W} = \mathbf{I} + \mathbf{L}(G)$, with $\mathbf{I}$ the identity matrix. As previously, W^{ij} denotes the cofactor of the w_{ij} element of $\mathbf{W}$. We will denote the set of the spanning rooted forest of a weighted multigraph G by $\mathcal{F}$, and $\mathcal{F}^{ij}$ designates the set of those spanning rooted forests of G where i and j belong to the same tree rooted at i .

Lemma 3.1 (Matrix-Forest Lemma for multigraphs). *For any weighted multigraph G ,*

1. $\det(\mathbf{W}) = \epsilon(\mathcal{F})$;
2. for any $i, j \in V(G)$, $W^{ij} = \epsilon(\mathcal{F}^{ij})$.

The first point in Lemma 3.1 expresses that the determinant of the $\mathbf{W}$ matrix is equal to the total weight of spanning rooted forests in the multigraph G , while the second one indicates that any cofactor of $\mathbf{W}$ represents the weight of the spanning rooted forests whose components are rooted in i and include j .

In the case of multidigraphs, the set of spanning diverging forests of G is denoted by $\mathcal{F}$, and $\mathcal{F}^{i \rightarrow j}$ is the set of those spanning diverging forests in which i and j belong to the same tree diverging from i :

Lemma 3.2 (Matrix-Forest Lemma for multidigraphs). *For any weighted multidigraph G ,*

1. $\det(\mathbf{W}) = \epsilon(\mathcal{F})$;
2. for any $i, j \in V(G)$, $W^{ij} = \epsilon(\mathcal{F}^{i \rightarrow j})$.

Let us denote the inverse of $\mathbf{W}$, $\mathbf{W}^{-1}$, by $\mathbf{Q} = (q_{ij}) = \mathbf{W}^{-1} = (\mathbf{I} + \mathbf{L})^{-1}$. $\mathbf{Q}$ is then equal to $\det(\mathbf{W})^{-1} \mathbf{W}^*$, with $\mathbf{W}^* = (W^{ij})^T$. This Lemma leads to the matrix-forest theorem :

Theorem 3.3 (Matrix-Forest Theorem).

1. For any weighted multigraph G , the matrix $\mathbf{Q} = \mathbf{W}^{-1}$ exists and $q_{ij} = \frac{\epsilon(\mathcal{F}^{ij})}{\epsilon(\mathcal{F})}$, $i, j = 1, \dots, n$.

2. For any weighted multidigraph G , the matrix $\mathbf{Q} = \mathbf{W}^{-1}$ exists and $q_{ij} = \frac{\epsilon(\mathcal{F}^{j \rightarrow i})}{\epsilon(\mathcal{F})}$, $i, j=1, \dots, n$.

The $\mathbf{Q}$ matrix can be considered as the *relative forest accessibilities* matrix between every pair of vertices of G , which can be seen as a similarity (or proximity) measure between vertices. Indeed, the more two vertices appear on the same tree in many forests, the more similar (or closer) they are likely to be. Chebotarev also shows that the $\mathbf{Q}$ matrix is doubly stochastic (when the graphs are undirected), which leads to the following interpretation of the elements of $\mathbf{Q}$: “ q_{ij} may be interpreted as the fraction of connectivity of vertices i and j in the total connectivity of i with all vertices” [24].

3.3 A new forest similarity measure

We will now extend the forest similarity measure defined above in order to assign a probability distribution on the set of forests present on a directed graph G (if the graph is undirected, it is assumed that, for each arc, there exist directed links in the two directions $i \rightarrow j$ and $j \rightarrow i$). To this end, let us define the set of rooted forests φ that can be defined in the graph G as $\mathcal{F} = \{\varphi_1, \varphi_2, \dots\}$. Intuitively, in the undirected case, a rooted forest is an acyclic subgraph of G that has the same nodes as G and one marked node (a root) in each component (see [23, 24] for details). In the directed case, diverging forests are considered, that is, forests containing diverging rooted trees (i.e., trees that contain only directed paths from the root to all the other nodes). Now, as we are dealing with directed graphs, diverging rooted trees and forests will simply be referred to as trees and forests. The *total cost* of a forest φ is defined as the *sum* of the individual costs of the arcs belonging to φ , $C(\varphi)$, while the *total weight* of such a forest φ is defined as the *product* of the individual weights (the elements of the adjacency matrix) of the arcs belonging to φ . A forest with no arc (containing only individual nodes without any connection) has a 0 total cost and a total weight of 1.

A **Boltzmann probability distribution** is defined on the set $\mathcal{F}$:

$$P(\varphi) = \frac{\exp[-\theta C(\varphi)]}{\sum_{\varphi' \in \mathcal{F}} \exp[-\theta C(\varphi')]} \quad (3.29)$$

where θ is a smoothing parameter related to the temperature in statistical physics. Thus, as expected, low-cost forests φ (having small $C(\varphi)$) are favored in that they have a large probability of being chosen. Indeed, from Equation (4.10), we clearly observe that when $\theta \rightarrow 0^+$, the forest probabilities tend to a uniform probability. On the other hand, when θ is large, the probability distribution defined by Equation (4.10) is biased towards low-cost

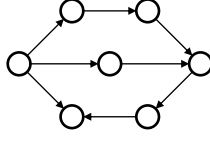


Figure 3.3: A directed graph G in which arc costs are uniformly 1.

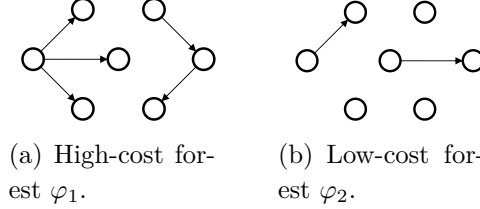


Figure 3.4: Examples of forests on graph G containing two trees.

forests (the most likely forests are the lowest-cost ones). Notice that in Equation (4.10) isolated nodes do not contribute to the probability mass, as they have no incoming or outgoing links.

For illustration, the simple graph G shown in Figure 3.3 is analysed. Figure 3.4 represents examples of respectively a high-cost forest φ_1 and a low-cost forest φ_2 of G . The cost associated to φ_1 is 5, as this forest contains five arcs with a cost equal to 1. Similarly, the cost of φ_2 is 2. The numerator of the Equation (4.10) for φ_1 becomes $\exp[-\theta 5]$, the numerator for φ_2 $\exp[-\theta 2]$, while the denominator is the same for both forests. For small values of θ , those numerators tend to 1, resulting in a uniform distribution. For high values of θ , the probability of the lower-cost forest φ_2 is significantly higher than the probability of the higher-cost forest φ_1 .

Let us now apply the same formalism as in Section 3.2 to define a new similarity measure between nodes of a graph. A new matrix $\mathbf{W}$ is derived from the cost matrix $\mathbf{C}$,

$$\mathbf{W} = \exp[-\theta \mathbf{C}], \quad (3.30)$$

and we define a new forest similarity, which is called the Bag-of-Forests (BoF) similarity (as we consider a bag containing all the forests of the graph, and picking forests at random), by

$$\mathbf{K}_{\text{BoF}} = (\mathbf{I} + \mathbf{L}(\mathbf{W}))^{-1}, \quad (3.31)$$

where $\mathbf{L}(\mathbf{W}) = \mathbf{D}_w - \mathbf{W}$ is the Laplacian matrix computed from $\mathbf{W}$ and $\mathbf{D}_w = \text{Diag}(\mathbf{W}\mathbf{e})$.

This similarity measure between two nodes i and j introduced here, which is an extension of the forest similarity, is a valid kernel and has a nice probabilistic interpretation: it is the probability for node i to be present on a tree rooted in j (and therefore connected to j , see [23]), knowing that forests are sampled according to a Boltzmann distribution.

It therefore corresponds to the *a posteriori probability* that the root node of i is j , given the leaf node i of interest.

As this similarity measure requires the inversion of the matrix $\mathbf{I} + \mathbf{L}(\mathbf{W})$ which has a time complexity of $\mathcal{O}(n^3)$, the overall time complexity of the $\mathbf{K}_{\text{BoF}}$ kernel is also in $\mathcal{O}(n^3)$.

Two other distances, and hence similarity measures, are defined below: the Bag-of-Forests potential (BoFPot) and logarithmic (BoFLog) distances.

3.3.1 The BoF potential distance

Following arguments detailed in [52], we define a new function $\phi_{ij} = -\frac{1}{\theta} \log(k'_{ij})$, called the *potential* where θ is the parameter of the similarity measure K_{BoF} , and $k'_{ij} = k_{ij}/k_{jj}$ is the i, j element of the matrix K_{BoF} , divided by the corresponding diagonal element. A distance is then defined by $\Delta_{ij} = (\phi_{ij} + \phi_{ji})/2$. The definition of the BoF potential distance is therefore

$$\Delta_{ij}^{\text{Pot}} = \begin{cases} (\phi_{ij} + \phi_{ji})/2 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}. \quad (3.32)$$

3.3.2 The BoF logarithmic distance

Similarly to [21], we define the matrix $\mathbf{D}$ by

$$\mathbf{D} = \ln(\mathbf{K}_{\text{BoF}}), \quad (3.33)$$

and the matrix Δ^{Log} by

$$\Delta^{\text{Log}} = \frac{1}{2}(\mathbf{d}\mathbf{1}^T + \mathbf{1}\mathbf{d}^T) - \mathbf{D}, \quad (3.34)$$

where $\mathbf{d}$ is the column vector containing the diagonal entries of $\mathbf{D}$, $\mathbf{d}^T$ is the transpose of $\mathbf{d}$, $\mathbf{1}$ and $\mathbf{1}^T$ being a column vector of n ones and its transpose. This is a classical transformation used to obtain a metric from a symmetric similarity measure (see for instance [33]).

Centered kernel matrices $\mathbf{K}_{\text{BoFPot}}$ and $\mathbf{K}_{\text{BoFLog}}$ can be derived from the Δ distance matrices using [29]

$$\mathbf{K}_{\text{BoFPot}} = -\frac{1}{2}\mathbf{H}\Delta^{\text{Pot}}\mathbf{H}, \quad (3.35)$$

$$\mathbf{K}_{\text{BoFLog}} = -\frac{1}{2}\mathbf{H}\Delta^{\text{Log}}\mathbf{H}, \quad (3.36)$$

where $\mathbf{H} = \mathbf{I} - \frac{(\mathbf{e}\mathbf{e}^T)}{n}$, n being the size of the matrix $\mathbf{H}$.

Those two similarity matrices will be used hereunder in a semi-supervised classification task, along with the original $\mathbf{K}_{BoF}$ kernel matrix, and other state of the art methods.

3.4 Experiments

This section is based on the same framework as [86].

Our goal in this section is to illustrate the behavior of our similarity measures $\mathbf{K}_{BoF}$ and its derivatives $\mathbf{K}_{BoFPot}$ and $\mathbf{K}_{BoFLog}$, compared to classical similarity measures, using a non trivial machine learning task, namely the semi-supervised classification of graph nodes, which is a special case of classical classification (supervised learning). As labelled data is often much harder to obtain the unlabelled data, semi-supervised classification uses those two kind of data to generate accurate classifiers, contrarily to supervised learning where only labelled data is used. Among all semi-supervised classification approaches, graph-based methods display good performance, as they allow to not only use information on the labels, but also the topology of the graph to attribute a label to an unknown node. The field of graph-based semi-supervised classification has received a lot of attention recently, and we refer the interested reader to existing reviews (see for instance [2, 20, 180, 181]).

Our main objective here is thus to infer labels for the unlabeled nodes from the labeled ones, that is, to classify unlabeled nodes on moderate to medium sized datasets.

3.4.1 Datasets

Newsgroup

Graphs generated from the Newsgroup dataset [85] are used. This dataset is originally composed of about 20,000 unstructured documents, taken from 20 discussion groups (newsgroups) of the Usenet diffusion list, and composed of 20 classes. For our experiments, three subsets related to different topics are extracted from the original database (NewsGroup1, 2, and 3) [173]. The graphs of documents were built by sampling at random about 200 documents in each of three classes from three different topics (see Table 3.1).

Class	NG1	NG2	NG3	NG4	NG5	NG6	NG7	NG8	NG9
1	200	198	200	200	200	197	200	200	200
2	200	200	199	200	198	200	200	200	200
3				200	200	198	200	198	197
4							200	200	200
5							198	200	200
Total	400	398	399	600	598	595	998	998	997

Table 3.1: Composition of the nine Newsgroup datasets.

Class	Cornell	Texas	Washington	Wisconsin
Course	54	51	170	83
Department	25	36	20	37
Faculty	62	50	44	37
Project	54	28	39	25
Staff	6	6	10	11
Student	145	163	151	155
Total	346	334	434	348

Table 3.2: Composition of the WebKB cocite datasets.

WebKB cocite

This dataset [101, 178] consists of four sets of web pages collected from four university departments, with each page manually labeled as belonging to one of six authorship categories: course, department, faculty, project, staff, and student. Those pages are linked by co-citation (x and y are linked if x links to z and y links to z)(see Table 3.2).

3.4.2 Experimental setup

Five values of the labeling rate (i.e., the proportion of nodes with known labels) were used (ranging from 10% to 90%) in the experiments. In order to assess the ability of the algorithms to determine correct labels, some labels were removed from the original dataset and the removed data was used as a test set (the remaining data constituting the training set). Indeed, a double cross-validation using a 10-fold external validation plus a 10-fold internal validation were used. The internal validation is performed in order to tune the parameter of the different methods used. This double cross-validation procedure was repeated ten times (for each method and each labelling rate), and the average classification rate was reported. Four different similarity measures (already described in section

3.1.3) were used to compare to our $\mathbf{K}_{BoF}$ measures: the regularized commute-time kernel ($\mathbf{K}_{RCT}$), the random-walk with restart similarity ($\mathbf{K}_{RWR}$), the regularized Laplacian kernel ($\mathbf{K}_{RL}$) and the bag-of-paths (BoP) betweenness.

For the three kernels cited above and the BoF kernels (i.e., all methods except the BoP), the classification of unlabeled nodes is performed with a simple kernel-alignment approach, based on a sum of similarities. More precisely, each class c and the nodes belonging to that class are described by an n -dimensional indicator vector $\mathbf{y}^c$ containing a “1” entry when the corresponding node belongs to class c and “0” otherwise (when the node is unlabeled or belongs to another class). The similarity of node i with the set of nodes belonging to class c is the value of the i th-entry of the column vector $\mathbf{K}\mathbf{y}^c$ where $\mathbf{K}$ is the kernel matrix. This similarity is computed for each class c . Each node is then assigned to the class with largest similarity. The column vector containing the estimated class index of all nodes is thus $\mathbf{y}^* = \operatorname{argmax}_c(\mathbf{K}\mathbf{y}^c)$.

For the bag-of-paths betweenness, a group betweenness is derived from the classical betweenness [86], allowing to compute the betweenness of a node i related to all the other nodes from a class. This gives an indication to which extent this node is in between all the other nodes from the class, and thus if node i is “in the neighborhood” of all the nodes of the class. This operation is repeated for each class, and the node i is assigned to the class showing the highest betweenness.

3.4.3 Results

In this section we present results obtained on a semi-supervised classification task.

The different curves shown in Figures 3.5, 3.6, 3.7, and 3.8 represent the classification rate (averaged on the 10 runs) of the nodes’ label (i.e., if the label predicted with the classification corresponds to the original one) for the five labelling rates, and interpolated between them. This gives us a nice visual approach of the results.

We also used two quantitative procedures to interpret the results. The first one is to compute one-sided t -tests for each pair of methods to determine if one is significantly superior to another, i.e., if the p -value is less than 0.05 on the 10 runs. The results of those t -tests are displayed in Table 3.3, 3.4, and 3.5. Each row of those tables shows how many times the method corresponding to the row is significantly superior to the method corresponding to the column (for instance, in the first line of the Table 3.3, we can see that for a labelling rate of 90%, the BoF method is significantly better than the BoFPot and the BoFLog in four cases, and better than the RL and the BoP methods in one case, on the NewsGroup datasets). We can also see in Tables 3.6 to 3.14 the Win/Tie/Lose frequencies of the BoF methods against all other methods. For example, in Table 3.14,

the BoFLog similarity measure shows 8 win, 1 tie and 4 lose against the original BoF measure, on all datasets.

The second one is a Borda ranking procedure [31, 130] that has been used to determine an overall ranking of the different methods. This procedure works as follow: N being the number of methods, for fixed dataset and labelling rate, N points are attributed to the “winner” of the classification task, i.e., the method presenting the best (averaged on the 10 runs) classification rate. $N - 1$ points are attributed to the second best method, and so on. A partial ranking is then constructed for each dataset and labelling rate, and those partial rankings are then summed over all the desired datasets (let’s say the NewsGroup ones). The method obtaining the higher score is the winner on those datasets, the second best is ranked second, etc. The results of the Borda procedure are displayed in Tables 3.15, 3.16, and 3.17.

In the quantitative procedures, the results are split into three cases, the results of the BoF methods being quite different from one kind of dataset to another, as detailed hereunder.

When looking at the results on the NewsGroup datasets only (Figures 3.5, 3.6, and 3.7), we immediately see that the RL method, corresponding to the “Forest Accessibility” measure from Chebotarev [21], and the BoF similarity measure are clearly below the other methods, which is confirmed by the Table 3.15, where they are ranked respectively seventh and eighth. For high labelling rates, the two other measures derived from the BoF similarity, BoFPot and BoFLog are generally situated in the middle of the pack, sometimes giving best results (see NewsGroup3 3.5(c)) and sometimes giving results just above the ones of the BoF (see NewsGroup8 3.7(b)). The classification rate is nevertheless greatly increased when going from the original BoF measure to the modified ones. The methods achieving best results for these datasets are RWWR, RCT and BoP.

Contrarily, when dealing with the WebKB Cocite datasets, the situation is quite different. This time the BoFPot and the BoFLog are among the methods giving best classification scores, and thus for every labelling rate, as we can see on Figure 3.8. RCT is however very close behind, being even better on one dataset (see Figure 3.8(d)). The Borda ranking confirms these results, placing BoFPot at the first place and the BoFLog at the second. BoF, RL are still at the end of the ranking, joined this time by the RWWR method. BoP and DW2 are in the middle of the ranking.

When grouping all datasets, as usual BoF and RL are ranked last. BoFPot is ranked third for low labelling rates, and fifth for higher labelling rates, while the ranking of BoFLog lies between fourth and seventh. RCT, BoP and RWWR are usually classified in the top three. DW2 shows good results when the number of furnished labels is high, but quickly drowns to the sixth place when the labelling becomes rare.

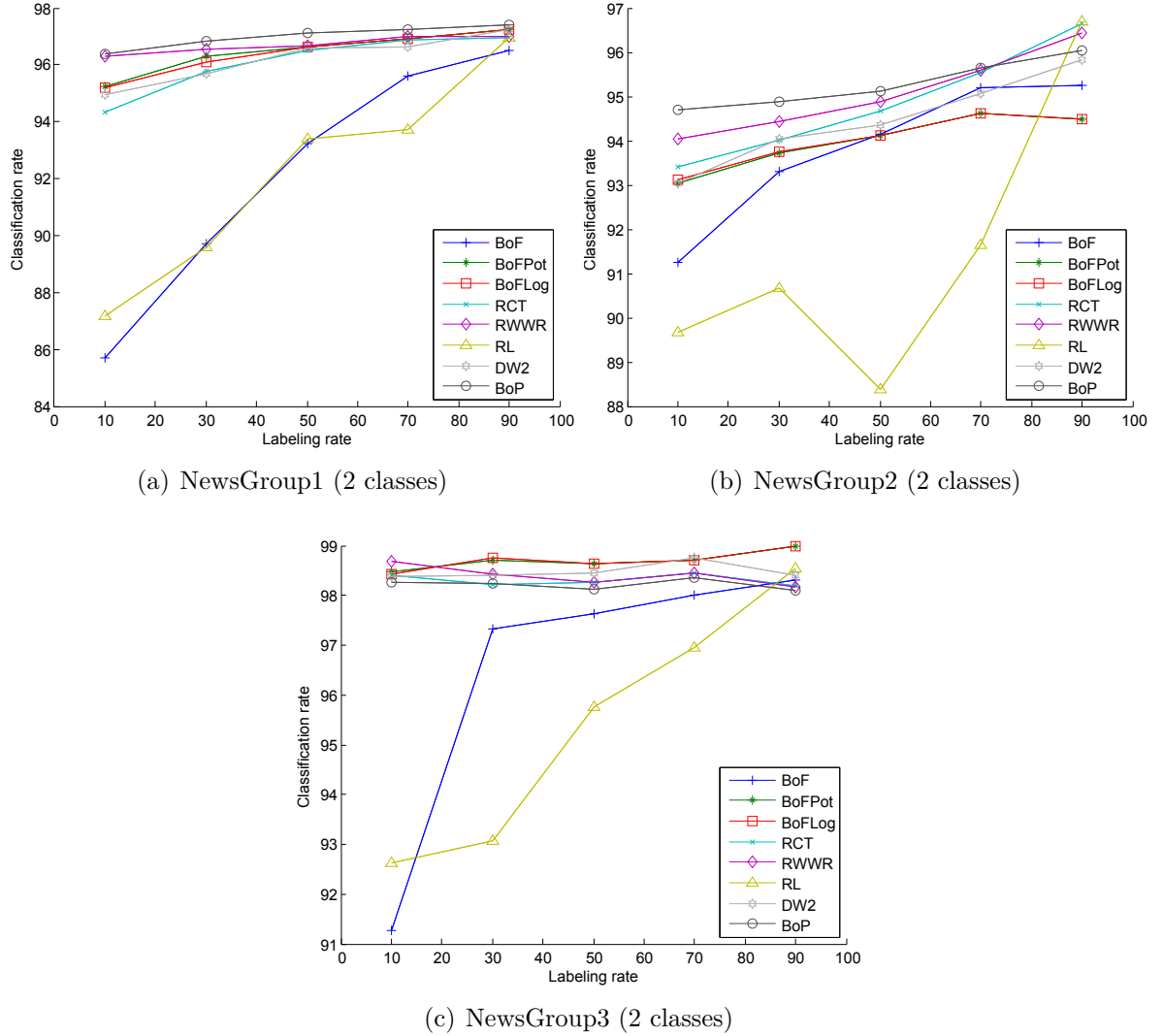


Figure 3.5: Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 2 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).

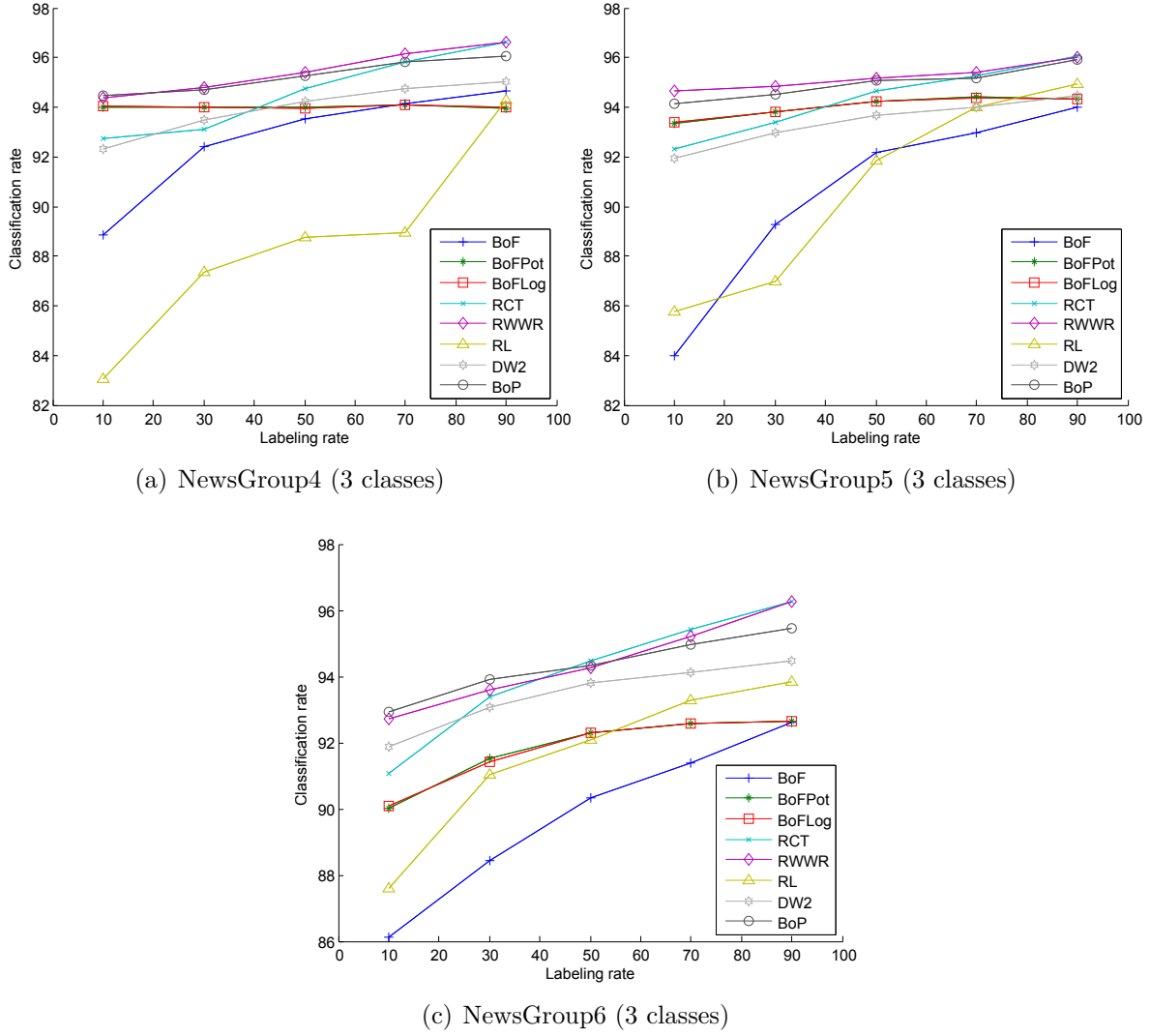


Figure 3.6: Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 3 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).

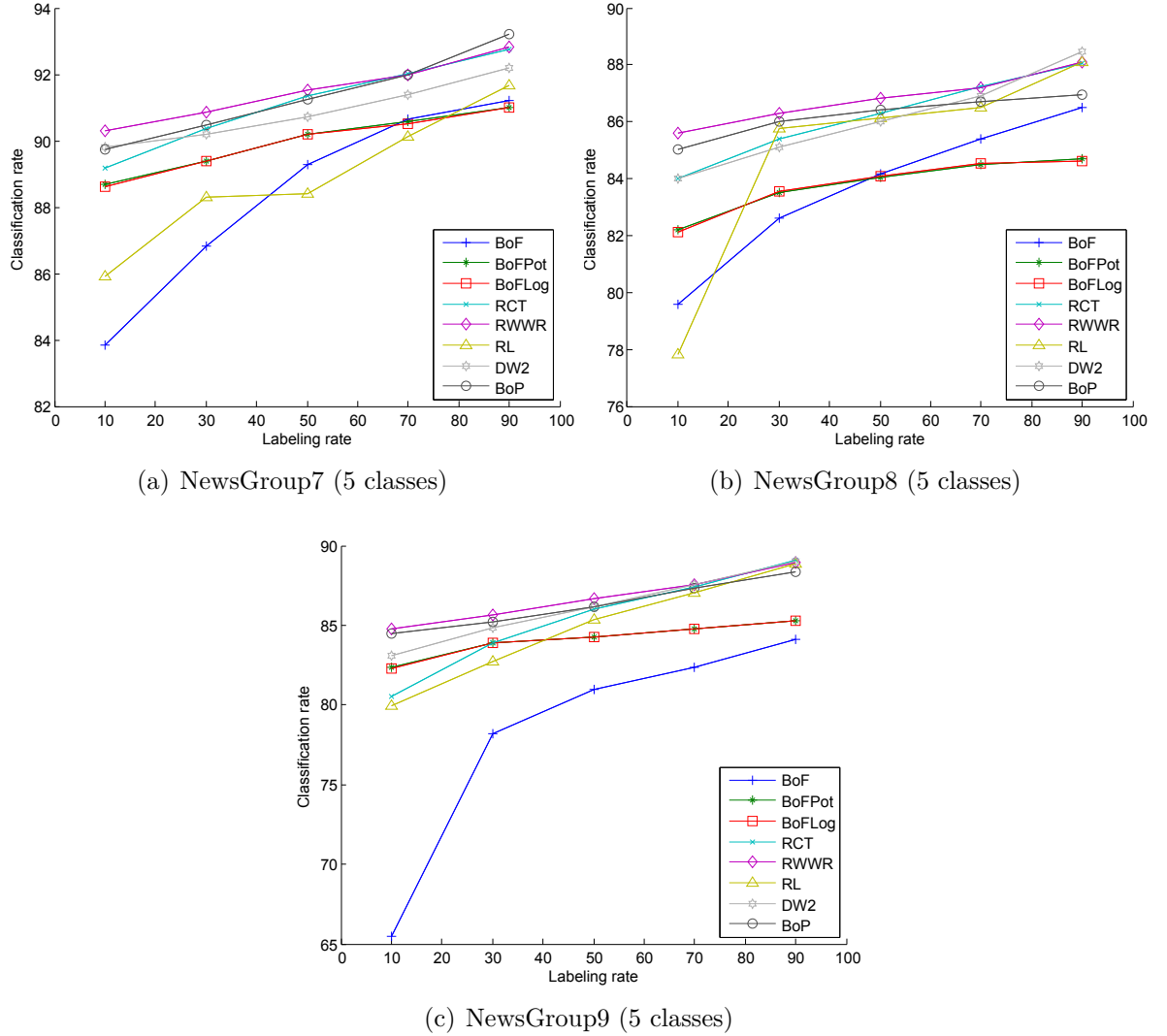


Figure 3.7: Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (NewsGroup with 5 classes). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).

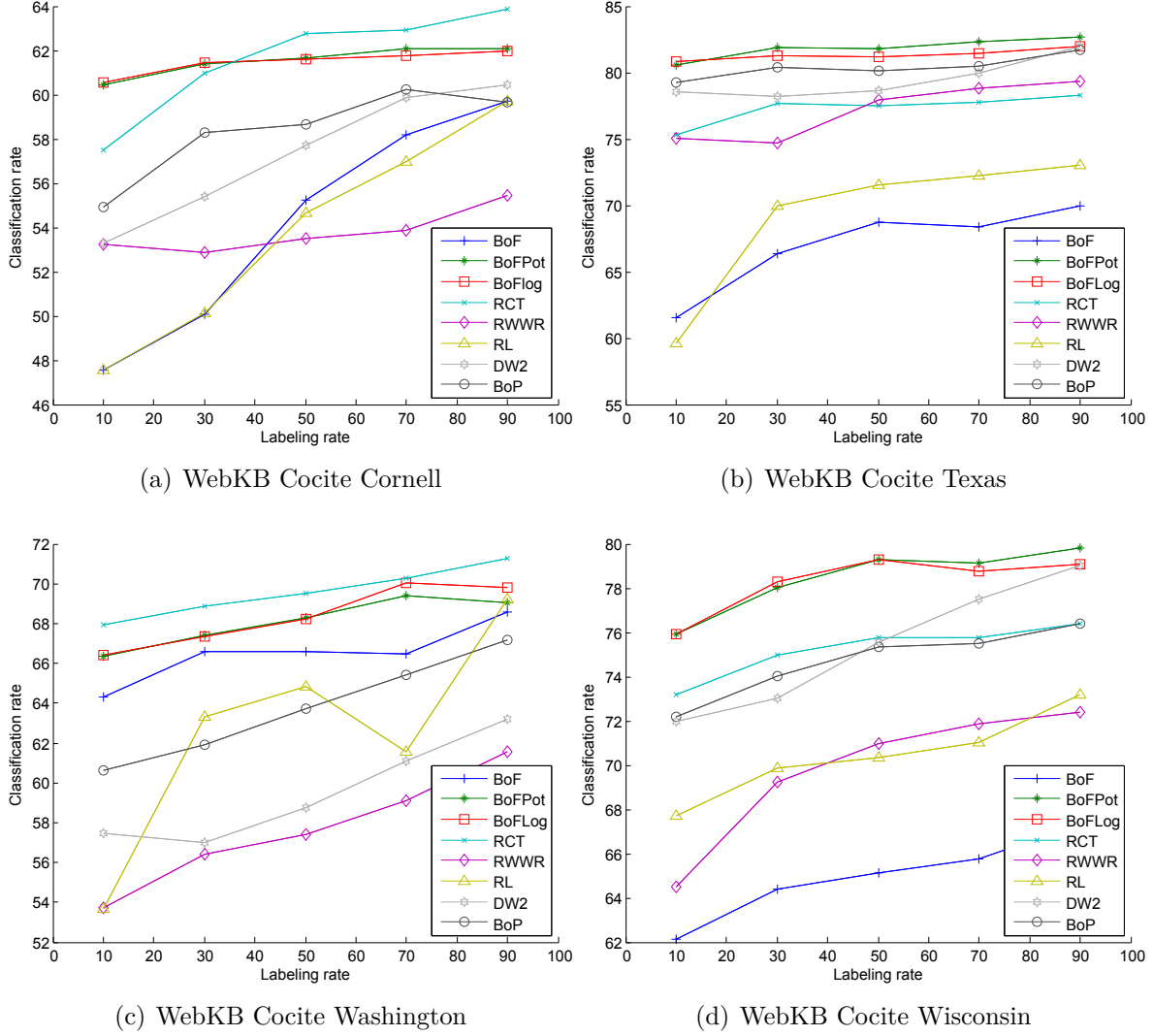


Figure 3.8: Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs (WebKB). Results are reported for eight methods (BoF, BoFPot, BoFLog, RCT, RWWR, RL, DW2, BoP) and five labelling rates (10%, 30%, 50%, 70%, 90%).

		BoF	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
Labelling rate: 90%	BoF	0	4	4	0	0	1	0	1
	BoFPot	4	0	1	2	2	2	1	1
	BoFLog	4	0	0	2	2	2	1	1
	RCT	8	7	7	0	1	5	5	5
	RWWR	8	7	7	0	0	4	5	5
	RL	8	7	7	1	2	0	2	4
	DW2	8	7	7	3	3	6	0	3
	BoP	8	8	8	2	2	5	5	0
Labelling rate: 70%	BoF	0	3	3	0	0	2	1	0
	BoFPot	5	0	3	1	1	5	2	1
	BoFLog	5	1	0	1	1	5	2	1
	RCT	9	7	7	0	1	9	7	2
	RWWR	9	8	8	4	0	9	7	5
	RL	4	3	3	0	0	0	0	0
	DW2	8	6	6	2	1	7	0	2
	BoP	9	8	8	1	1	8	6	0
Labelling rate: 50%	BoF	0	0	0	0	0	3	0	0
	BoFPot	7	0	1	1	1	6	2	1
	BoFLog	7	0	0	1	1	7	2	1
	RCT	9	7	7	0	1	9	6	1
	RWWR	9	7	7	7	0	9	8	4
	RL	3	2	2	0	0	0	1	0
	DW2	9	6	6	2	1	8	0	1
	BoP	9	8	8	5	2	9	7	0
Labelling rate: 30%	BoF	0	0	0	0	0	3	0	0
	BoFPot	9	0	1	4	1	8	4	1
	BoFLog	9	0	0	4	1	8	4	1
	RCT	9	4	3	0	0	8	4	0
	RWWR	9	8	8	9	0	9	9	5
	RL	4	1	1	1	0	0	1	0
	DW2	9	5	5	3	0	8	0	1
	BoP	9	8	8	7	3	9	8	0
Labelling rate: 10%	BoF	0	0	0	0	0	1	0	0
	BoFPot	9	0	0	4	0	9	2	0
	BoFLog	9	0	0	4	0	9	2	0
	RCT	9	4	3	0	0	8	3	1
	RWWR	9	8	8	9	0	9	9	5
	RL	3	0	0	0	0	0	0	0
	DW2	9	4	4	4	0	9	0	0
	BoP	9	8	8	8	1	9	7	0

Table 3.3: One-side t -test for every labelling rate for NewsGroup datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.

		BoF	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
Labelling rate: 90%	BoF	0	0	0	0	2	0	1	1
	BoFPot	4	0	2	2	4	3	4	4
	BoFLog	4	1	0	2	4	4	2	3
	RCT	4	2	2	0	3	4	2	2
	RWWR	2	0	0	1	0	1	0	0
	RL	3	0	0	0	3	0	1	1
	DW2	3	0	0	2	4	3	0	1
	BoP	2	0	0	1	4	2	1	0
Labelling rate: 70%	BoF	0	0	0	0	2	2	1	1
	BoFPot	4	0	3	2	4	4	4	4
	BoFLog	4	1	0	2	4	4	4	4
	RCT	4	2	1	0	3	4	2	3
	RWWR	2	0	0	1	0	2	0	0
	RL	2	0	0	0	2	0	0	0
	DW2	3	0	0	2	4	3	0	1
	BoP	3	0	0	1	4	4	2	0
Labelling rate: 50%	BoF	0	0	0	0	2	1	1	1
	BoFPot	4	0	1	2	4	4	4	4
	BoFLog	4	0	0	2	4	4	4	4
	RCT	4	2	2	0	3	4	2	3
	RWWR	2	0	0	1	0	2	0	0
	RL	2	0	0	0	2	0	1	1
	DW2	3	0	0	1	4	3	0	0
	BoP	3	0	0	1	4	3	2	0
Labelling rate: 30%	BoF	0	0	0	0	1	1	1	1
	BoFPot	4	0	1	2	4	4	4	4
	BoFLog	4	1	0	2	4	4	4	4
	RCT	4	1	1	0	4	4	3	3
	RWWR	3	0	0	0	0	2	0	0
	RL	2	0	0	0	2	0	1	1
	DW2	3	0	0	1	4	3	0	0
	BoP	3	0	0	1	4	3	4	0
Labelling rate: 10%	BoF	0	0	0	0	1	1	1	1
	BoFPot	4	0	0	3	4	4	4	4
	BoFLog	4	0	0	3	4	4	4	4
	RCT	4	1	1	0	3	4	3	3
	RWWR	3	0	0	0	0	2	0	0
	RL	1	0	0	0	1	0	0	0
	DW2	3	0	0	1	3	4	0	0
	BoP	3	0	0	1	4	4	2	0

Table 3.4: One-side t -test for every labelling rate for WebKB datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.

		BoF	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
Labelling rate: 90%	BoF	0	4	4	0	2	1	1	2
	BoFPot	8	0	3	4	6	5	5	5
	BoFLog	8	1	0	4	6	6	3	4
	RCT	12	9	9	0	4	9	7	7
	RWWR	10	7	7	1	0	5	5	5
	RL	11	7	7	1	5	0	3	5
	DW2	11	7	7	5	7	9	0	4
	BoP	10	8	8	3	6	7	6	0
Labelling rate: 70%	BoF	0	3	3	0	2	4	2	1
	BoFPot	9	0	6	3	5	9	6	5
	BoFLog	9	2	0	3	5	9	6	5
	RCT	13	9	8	0	4	13	9	5
	RWWR	11	8	8	5	0	11	7	5
	RL	6	3	3	0	2	0	0	0
	DW2	11	6	6	4	5	10	0	3
	BoP	12	8	8	2	5	12	8	0
Labelling rate: 50%	BoF	0	0	0	0	2	4	1	1
	BoFPot	11	0	2	3	5	10	6	5
	BoFLog	11	0	0	3	5	11	6	5
	RCT	13	9	9	0	4	13	8	4
	RWWR	11	7	7	8	0	11	8	4
	RL	5	2	2	0	2	0	2	1
	DW2	12	6	6	3	5	11	0	1
	BoP	12	8	8	6	6	12	9	0
Labelling rate: 30%	BoF	0	0	0	0	1	4	1	1
	BoFPot	13	0	2	6	5	12	8	5
	BoFLog	13	1	0	6	5	12	8	5
	RCT	13	5	4	0	4	12	7	3
	RWWR	12	8	8	9	0	11	9	5
	RL	6	1	1	1	2	0	2	1
	DW2	12	5	5	4	4	11	0	1
	BoP	12	8	8	8	7	12	12	0
Labelling rate: 10%	BoF	0	0	0	0	1	2	1	1
	BoFPot	13	0	0	7	4	13	6	4
	BoFLog	13	0	0	7	4	13	6	4
	RCT	13	5	4	0	3	12	6	4
	RWWR	12	8	8	9	0	11	9	5
	RL	4	0	0	0	1	0	0	0
	DW2	12	4	4	5	3	13	0	0
	BoP	12	8	8	9	5	13	9	0

Table 3.5: One-side t -test for every labelling rate for all datasets. Each entry of the table shows on how many datasets row method is significantly better than column method.

Labelling rate	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	4/1/4	4/1/4	0/1/8	0/1/8	1/0/8	0/1/8	1/0/8
70%	3/1/5	3/1/5	0/0/9	0/0/9	2/3/4	1/0/8	0/0/9
50%	0/2/7	0/2/7	0/0/9	0/0/9	3/3/3	0/0/9	0/0/9
30%	0/0/9	0/0/9	0/0/9	0/0/9	3/2/4	0/0/9	0/0/9
10%	0/0/9	0/0/9	0/0/9	0/0/9	1/5/3	0/0/9	0/0/9

Table 3.6: Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on NewsGroup datasets.

Labelling rate	BoF	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	4/1/4	1/8/0	2/0/7	2/0/7	2/0/7	1/1/7	1/0/8
70%	5/1/3	3/5/1	1/1/7	1/0/8	5/1/3	2/1/6	1/0/8
50%	7/2/0	1/8/0	1/1/7	1/1/7	6/1/2	2/1/6	1/0/8
30%	9/0/0	1/8/0	4/1/4	1/0/8	8/0/1	4/0/5	1/0/8
10%	9/0/0	0/9/0	4/1/4	0/1/8	9/0/0	2/3/4	0/1/8

Table 3.7: Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on NewsGroup datasets.

Labelling rate	BoF	BoFPot	RCT	RWWR	RL	DW2	BoP
90%	4/1/4	0/8/1	2/0/7	2/0/7	2/0/7	1/1/7	1/0/8
70%	5/1/3	1/5/3	1/1/7	1/0/8	5/1/3	2/1/6	1/0/8
50%	7/2/0	0/8/1	1/1/7	1/1/7	7/0/2	2/1/6	1/0/8
30%	9/0/0	0/8/1	4/2/3	1/0/8	8/0/1	4/0/5	1/0/8
10%	9/0/0	0/9/0	4/2/3	0/1/8	9/0/0	2/3/4	0/1/8

Table 3.8: Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on NewsGroup datasets.

Labelling rate	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	0/0/4	0/0/4	0/0/4	2/0/2	0/1/3	1/0/3	1/1/2
70%	0/0/4	0/0/4	0/0/4	2/0/2	2/0/2	1/0/3	1/0/3
50%	0/0/4	0/0/4	0/0/4	2/0/2	1/1/2	1/0/3	1/0/3
30%	0/0/4	0/0/4	0/0/4	1/0/3	1/1/2	1/0/3	1/0/3
10%	0/0/4	0/0/4	0/0/4	1/0/3	1/2/1	1/0/3	1/0/3

Table 3.9: Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on WebKB datasets.

Labelling rate	BoF	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	4/0/0	2/1/1	2/0/2	4/0/0	3/1/0	4/0/0	4/0/0
70%	4/0/0	3/0/1	2/0/2	4/0/0	4/0/0	4/0/0	4/0/0
50%	4/0/0	1/3/0	2/0/2	4/0/0	4/0/0	4/0/0	4/0/0
30%	4/0/0	1/2/1	2/1/1	4/0/0	4/0/0	4/0/0	4/0/0
10%	4/0/0	0/4/0	3/0/1	4/0/0	4/0/0	4/0/0	4/0/0

Table 3.10: Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on WebKB datasets.

Labelling rate	BoF	BoFPot	RCT	RWWR	RL	DW2	BoP
90%	4/0/0	1/1/2	2/0/2	4/0/0	4/0/0	2/2/0	3/1/0
70%	4/0/0	1/0/3	2/1/1	4/0/0	4/0/0	4/0/0	4/0/0
50%	4/0/0	0/3/1	2/0/2	4/0/0	4/0/0	4/0/0	4/0/0
30%	4/0/0	1/2/1	2/1/1	4/0/0	4/0/0	4/0/0	4/0/0
10%	4/0/0	0/4/0	3/0/1	4/0/0	4/0/0	4/0/0	4/0/0

Table 3.11: Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on WebKB datasets.

Labelling rate	BoFPot	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	4/1/8	4/1/8	0/1/12	2/1/10	1/1/11	1/1/11	2/1/10
70%	3/1/9	3/1/9	0/0/13	2/0/11	4/3/6	2/0/11	1/0/12
50%	0/2/11	0/2/11	0/0/13	2/0/11	4/4/5	1/0/12	1/0/12
30%	0/0/13	0/0/13	0/0/13	1/0/12	4/3/6	1/0/12	1/0/12
10%	0/0/13	0/0/13	0/0/13	1/0/12	2/7/4	1/0/12	1/0/12

Table 3.12: Table summarizing the Win/Tie/Lose frequencies of the BoF against every other method, on all datasets.

Labelling rate	BoF	BoFLog	RCT	RWWR	RL	DW2	BoP
90%	8/1/4	3/9/1	4/0/9	6/0/7	5/1/7	5/1/7	5/0/8
70%	9/1/3	6/5/2	3/1/9	5/0/8	9/1/3	6/1/6	5/0/8
50%	11/2/0	2/11/0	3/1/9	5/1/7	10/1/2	6/1/6	5/0/8
30%	13/0/0	2/10/1	6/2/5	5/0/8	12/0/1	8/0/5	5/0/8
10%	13/0/0	0/13/0	7/1/5	4/1/8	13/0/0	6/3/4	4/1/8

Table 3.13: Table summarizing the Win/Tie/Lose frequencies of the BoFPot against every other method, on all datasets.

Labelling rate	BoF	BoFPot	RCT	RWWR	RL	DW2	BoP
90%	8/1/4	1/9/3	4/0/9	6/0/7	6/0/7	3/3/7	4/1/8
70%	9/1/3	2/5/6	3/2/8	5/0/8	9/1/3	6/1/6	5/0/8
50%	11/2/0	0/11/2	3/1/9	5/1/7	11/0/2	6/1/6	5/0/8
30%	13/0/0	1/10/2	6/3/4	5/0/8	12/0/1	8/0/5	5/0/8
10%	13/0/0	0/13/0	7/2/4	4/1/8	13/0/0	6/3/4	4/1/8

Table 3.14: Table summarizing the Win/Tie/Lose frequencies of the BoFLog against every other method, on all datasets.

Labelling rate	1st	2nd	3rd	4th	5th	6th	7th	8th
90%	RCT	RWWR	DW2	BoP	RL	BoFPot	BoFLog	BoF
70%	RWWR	RCT	BoP	DW2	BoFPot	BoFLog	BoF	RL
50%	RWWR	BoP	RCT	DW2	BoFPot	BoFLog	BoF	RL
30%	RWWR	BoP	BoFPot	RCT	DW2	BoFLog	RL	BoF
10%	RWWR	BoP	BoFPot	BoFLog	RCT	DW2	RL	BoF

Table 3.15: Ranking of the different methods using a Borda ranking method on News-Group datasets and for all labelling rate.

Labelling rate	1st	2nd	3rd	4th	5th	6th	7th	8th
90%	BoFPot	BoFLog	RCT	DW2	BoP	RL	BoF	RWWR
70%	BoFPot	BoFLog	RCT	BoP	DW2	BoF	RWWR	RL
50%	BoFPot	BoFLog	RCT	BoP	DW2	BoF	RL	RWWR
30%	BoFPot	BoFLog	RCT	BoP	DW2	RL	RWWR	BoF
10%	BoFPot	BoFLog	RCT	BoP	DW2	RWWR	BoF	RL

Table 3.16: Ranking of the different methods using a Borda ranking method on WebKB datasets and for all labelling rate.

Labelling rate	1st	2nd	3rd	4th	5th	6th	7th	8th
90%	RCT	DW2	BoP	RWWR	BoFPot	RL	BoFLog	BoF
70%	RCT	BoP	RWWR	DW2	BoFPot	BoFLog	BoF	RL
50%	BoP	RCT	RWWR	BoFPot	BoFLog	DW2	BoF	RL
30%	BoP	RWWR	BoFPot	BoFLog	RCT	DW2	RL	BoF
10%	RWWR	BoP	BoFPot	BoFLog	RCT	DW2	RL	BoF

Table 3.17: Ranking of the different methods using a Borda ranking method on all datasets and for all labelling rate.

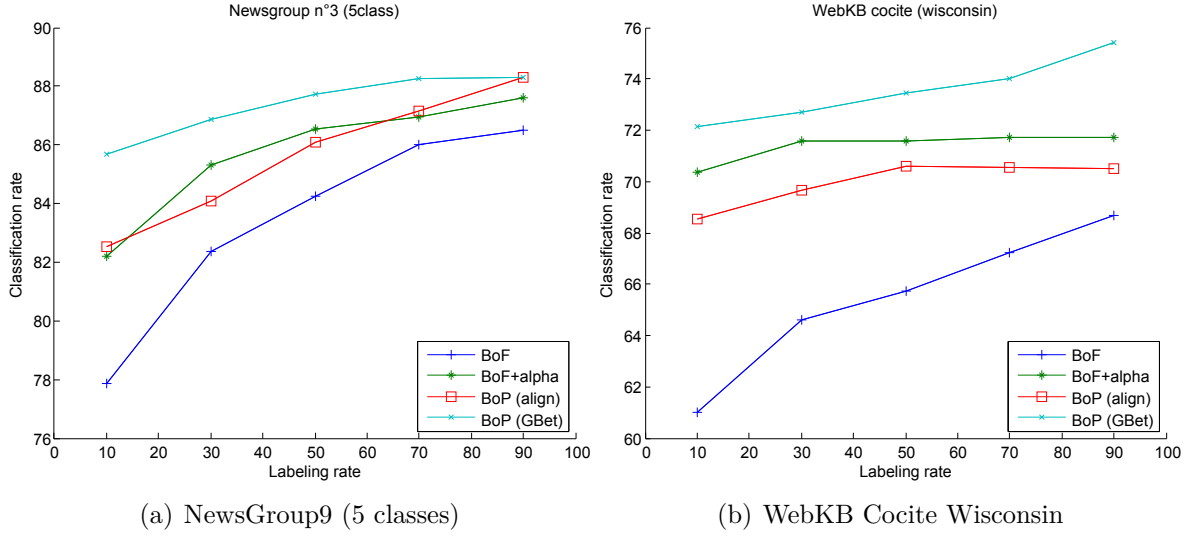


Figure 3.9: Classification rates in percent, averaged over 10 runs, obtained on partially labelled graphs. Results are reported for four methods (BoF, BoF+alpha, BoP with alignment (align), BoP with group betweenness (GBet)) and five labelling rates (10%, 30%, 50%, 70%, 90%).

Figure 3.9 illustrates more in details the behavior of the BoP kernel, based on paths, and the BoF kernel, based on forests. In addition to the previous experiments, and in order to compare adequately both kernels, the semi-supervised classification was computed using a kernel alignment for the BoP kernel (displayed as the “BoP (align)”) and compared to the BoF kernel also using an alignment. Another version of the BoF kernel, using a parameter α has also been tested:

$$\mathbf{K}_{\text{BoF}\alpha} = (\mathbf{I} + \alpha \mathbf{L}(\mathbf{W}))^{-1}. \quad (3.37)$$

When using the raw methods (simple BoF and BoP with alignment) the results displayed show an advantage of the BoP over the BoF. This may be due to the fact that conceptually, the probability of two nodes to be on the same path is lower than to be on a tree (as the path is more restrictive), and thus the similarity computed using forests may illustrate a weaker link than the BoP does. When the parameter α is introduced in the BoF kernel, the results show an interesting increase, surpassing the BoP (with alignment) on the two datasets used. In future research, this variation of the BoF kernel should however be studied further, and on a more complete group of datasets.

3.5 Chapter conclusion

In this chapter, new measures of similarity between nodes of graphs have been derived from the matrix-forest theorem [24, 25]. They are based on the sampling of forests on a graph, which are weighted by a Boltzmann probability distribution, assigning a greater probability of being picked to low-cost (“small”) forests. Two nodes are then similar if they appear together on many low-cost forests.

These similarity measures have been assessed on a semi-supervised classification task, trying to infer unknown labels of certain nodes from the knowledge of known labels on other nodes. Compared to state of the art methods, the K_{BoFPot} and K_{BoFLog} similarity measures perform in a very good way on some datasets (WebKB), and in an average way on other datasets (NewsGroup). In order to determine more precisely the situations where our measures perform well, tests on other datasets should be performed in the future.

Part II

Density measures on graphs

Chapter 4

The Sum-over-Forests density index

Contents

4.1	Review of density measures on graphs	73
4.1.1	Dense subgraphs	73
4.1.2	Local density scores	75
4.1.3	Smoothing of local measures	76
4.1.4	Global density scores	77
4.2	The SoF density index	77
4.3	Computation of the partition function $\mathcal{Z}$	78
4.4	Computation of the SoF density index	79
4.5	Experiments	80
4.5.1	Datasets	81
4.5.2	Graph construction	83
4.5.3	Evaluation methods	84
4.5.4	Results	85
4.6	Chapter conclusion	108

Extending the Bag-of-Forests similarities described before, our idea is to design a new density measure on graphs, also based on the matrix-forest theorem from Chebotarev. Indeed, density is an important concept in graph analysis and has been proven to be of particular interest in various areas such as, for example, social networks, biology and World-Wide-Web. Indeed, in social networks, identifying dense regions leads to the discovery of communities (group of people which is based on age, country, etc.) [3, 96]. In biology, finding dense components in protein-protein interaction networks (networks representing the interactions between proteins in some cellular process) helps to reveal

the function of those proteins [90, 121]. Concerning the World-Wide-Web graph, dense components can also be extracted, identifying for instance sets of web pages linked by a common topic [45, 83].

The task of identifying dense regions on a graph can be based on various concepts (degree of a node, cliques, cores, etc.) leading to diverse approaches. The key concept on which our approach is based is forest enumeration and, in particular, the matrix-forest theorem [23, 24], an extension of the well-known matrix-tree theorem (see, e.g., [157]). More precisely, the method developed in this chapter, inspired by [5, 107, 131, 174] (based on paths instead of forests), relies on the enumeration of all the possible forests in the graph, therefore leading to the definition of a new density index which will be called the **Sum-over-Forests (SoF) density index**. This measure has a clear and intuitive interpretation: when enumerating all the possible forests in the graph, a node will be considered as having a high density index if it is part of a tree of many – preferably low-cost – forests, and has a high outdegree within this forest. Indeed, if a region has a high density, it will contain a large number of trees – and therefore forests – so that the nodes belonging to that region will be part of many forests and have a high outdegree. Those nodes will thus obtain a high SoF density index.

In order to compute this index, we proceed in the same way as with the BoF similarity measures, first defining a Boltzmann probability distribution on the countable set of forests in the graph by adopting a statistical physics framework. This distribution has the desired property that high-cost forests occur with a low probability while low-cost forests occur with a high probability. As in statistical physics, it depends on a parameter, $\theta = 1/T$, controlling the temperature T – and thus the entropy – of the system. When T is low, only low-cost forests are taken into account (high-cost forest having a negligible contribution) while for high values of T , high-cost forests are as important as the low-cost ones (uniform distribution).

In a second step, the SoF density index of a node is defined according to this probability distribution. Roughly speaking, it corresponds to the expectation of the outdegree of this node, averaged over all the forests (the expectation is taken on all the possible forests). Technically speaking, the SoF density index is obtained by taking the first-order derivative of the partition function associated to the system. It is shown that it can be computed in closed form by inverting a $n \times n$ matrix depending on the immediate costs assigned to the arcs.

4.1 Review of density measures on graphs

In this section, we introduce a more formal definition of density on graphs, and review the different approaches used to identify dense areas (or components, or subgraphs) on graphs.

According to [87], two types of density can be defined: absolute and relative density. In the former, dense components are defined in a unique way, independently of what is outside the component. For instance, some methods are focused on finding cliques on graphs, which are completely connected subgraphs. In the latter, the density of an area on a graph is compared to the one of another area, aiming at finding the densest regions, where the number of regions to identify must sometimes be predefined. The identification of dense areas using the notion of relative density can be related to nodes clustering, and will be discussed further in Chapter 5.

We will now review a well-known approach for detecting high density regions on graphs relies on identifying dense, highly connected subgraphs like cliques, plexes, cores, etc. (see, e.g., [16]), which are measures originally conceived to evaluate cohesion in social groups [168].

4.1.1 Dense subgraphs

Cliques are completely connected subgraphs of the original graph [96]. In a way, they represent the densest possible area on graphs, an ideal to attain. However, the notion of clique is very restrictive: if an arc is missing, then the subgraph is no more considered as a clique. In real life data sets, this is of great importance because some edges could be missing either naturally or due to the collection of data presenting errors. Moreover, finding all the cliques, or the maximal clique in a graph is *NP-Complete*. A well-known exact algorithm for finding maximal cliques in an undirected graph is the one from Bron and Kerbosch [18]. It is based on an enumeration approach (it enumerates all possible subgraphs and check if some of them verify the definition of clique), which uses branch-and-bound techniques to prune branches which are unable to produce a clique. As in large applications it is excluded to enumerate all maximal cliques, there has also been extensive research on heuristic algorithms for solving maximum clique problems, based on methods including branch and bound [9], local search [78] and greedy algorithms [58]. In order to overcome the disadvantages of the excessively restrictive notion of clique, other structures relaxing this notion appeared. Those relaxations can be separated in three categories: density, degree and distance based.

γ -**quasi-clique**, or γ -clique [1], is a relaxed version of clique based on density. In-

deed, it is defined as a subgraph S which has an overall density of γ , i.e., S has at least $\gamma|S|(|S| - 1)/2$ edges, giving more flexibility when searching for dense components. Exact algorithms for finding maximal quasi-cliques exists, like the *Quick* algorithm which uses an efficient pruning technique [93], but the problem remains *NP-Complete* [120]. Abello [1] has provided a heuristic algorithm using a GRASP (greedy randomized adaptive search procedure) procedure. GRASP is an iterative method which, at each iteration, builds a randomized solution and then finds a locally optimal solution by searching the neighborhood of this solution. This approach which gives no guarantee about the quality of the solutions found, but is proved to be efficient in practice for many combinatorial optimization problems.

k-Clique [97] can be defined as a subgraph S whose diameter (the longest shortest-path between any two nodes) is smaller or equal to k . In other words, the shortest path from any vertex to any other vertex in S is not more than k . Note that a clique is equal to a k -clique with $k = 1$. The problem of enumerating k -cliques is not better than enumerating cliques and finding k -cliques remains intractable.

k-Plex [141] is degree-based relaxed version of a clique: it is defined as a subgraph containing n nodes where each node is connected to at least $n - k$ other nodes. Finding maximum k -plex is as hard as finding cliques [16]. Balasundaram et al. [10] formulated the maximum k -plex problem as an integer program and designed a branch-and-cut algorithm, while McClosky [109] adapted a combinatorial clique algorithms to solve this problem.

k-Cores (see, e.g., [12, 16]) are similar to plexes, but instead of specifying how many links are missing to produce a clique, nodes inside k -core only have to present a degree superior to k [117]. All nodes of the core are then connected to at least k other members of the core. More formally, let note the degree of a node i belonging to a subgraph H by $d_i(H)$. If the set of nodes of H is C , then a subgraph H_j generated by the set of nodes C_j is called a k -core if $d_i(H_j) \geq k \forall i \in C_j$. If H_j is a maximal subgraph with no larger subgraph H'_j containing it ($H_j \subset H'_j$), then H_j is a maximal k -core. Larger values of k correspond to vertices with larger degree, thus occupying a more central position in the network. Contrarily to cliques and plexes, cores can be computed in polynomial time, and there even exists a linear-time algorithm computing the core structure of a network [12]: the idea is to iteratively remove all nodes having a degree strictly inferior to k , as well as nodes whose degree becomes inferior to k due to the deletion. Doing so until there are no more nodes having a degree lower than k leads to a maximal k -core. Each node is then assigned with a core number corresponding to the core it belongs to.

Generalized k-Cores Batagelj et al. [12] extended the classical k -core in order to use other node properties (instead of the degree) such as the in/out degree on a directed graph, the strength on a weighted graph, the clustering coefficient, etc. The generalized

k -core is then defined in the same way than the k -core, but uses a measure of a node i on a graph G , $\rho_i(G)$. Therefore, a subgraph H_j defined by the set of nodes C_j is called a generalized k -core of order k if $\rho_i(H_j) \geq k \forall i \in C_j$. The same simple algorithm used to find maximal k -cores in the classical case can also be used here [12], if the measure ρ is monotone. Nodes having a measure inferior to a threshold k are recursively removed, and the measure ρ is updated. The result is a maximal generalized k -core.

4.1.2 Local density scores

Another approach for finding dense zones is to compute a density index (or score) on the nodes of a graph. One of the most intuitive density index is the **degree** of a node (on undirected graphs, in/out degree on directed graphs) defined as the number of links a node has. Indeed, the larger the number of neighbors of a node, the higher the density around it. This measure is then purely local, taking only into account the direct neighbors. The **strength** of a node is an extension of the degree to weighted graphs, computing the sum of the weights borne by the arcs of the neighboring nodes. When those weights are all equal to one, the strength reduces to the degree.

4.1.2.1 Strength

The strength of a node is defined as the sum of the affinities beared by the edges between this node and its neighbors. It is a generalization of the degree to weighted graphs:

$$\text{Str} = \sum_{j \in N(i)} a_{ij}, \quad (4.1)$$

$N(i)$ being the neighborhood of node i .

4.1.2.2 Clustering Coefficient

Two types of clustering coefficient (CC) can be defined: a global and a local one. The global CC computes the ratio between the number of triangles in a graph G (closed triples) and the number of paths of length two connecting three nodes (connected triples) [50]:

$$\text{CC}(G) = \frac{3(\text{number of triangles in } G)}{\text{number of connected triples in } G}. \quad (4.2)$$

A local version of the clustering coefficient can be derived from this definition. For a node i in a graph G :

$$CC_i = \frac{\text{number of pairs of neighbors of } i \text{ that are connected}}{\text{number of pairs of neighbors of } i}. \quad (4.3)$$

The latter coefficient, was extended to weighted graphs in [11], and is used in this work as a measure of density on nodes.

4.1.3 Smoothing of local measures

As the different measures, like the strength and the CC, may have a high variability, it may be interesting to smooth them by taking into account the value of the measure on the neighboring nodes. Two smoothing methods are presented below: the Laplacian regularization and the PageRank-like smoothing [50].

4.1.3.1 Smoothing through Laplacian regularization

The Laplacian matrix of an (undirected weighted) graph is defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathbf{D}$ being the diagonal degree matrix and $\mathbf{A}$ being the adjacency matrix of the graph. This smoothing (the smoothed measure is noted $\hat{\rho}$) consists in minimizing an objective function

$$c(\hat{\rho}) = \underbrace{\frac{1}{2} \sum_{i=1}^n (\hat{\rho}_i - \rho_i)^2}_{\text{least square cost}} + \alpha \underbrace{\sum_{i,j=1}^n a_{ij} (\hat{\rho}_i - \hat{\rho}_j)^2}_{\text{regularization term}} \quad (4.4)$$

by finding a trade-off between not deviating too much from the actual observed value (least square cost term) and the value computed on the neighbors (regularization term) [50]. Then taking the partial derivatives of $c(\hat{\rho})$ with respect to $\hat{\rho}_k$ and rearranging the terms in order to introduce the Laplacian matrix gives

$$(\mathbf{I} + \alpha \mathbf{L}) \hat{\rho} = \rho, \quad (4.5)$$

which can be rewritten

$$\hat{\rho} = (\mathbf{I} + \alpha \mathbf{L})^{-1} \rho. \quad (4.6)$$

4.1.3.2 PageRank-like smoothing

The smoothing is simply a weighted average between the value of the measure on a node and the values taken by its neighbors:

$$\hat{\rho}_i = (1 - \alpha) \rho_i + \alpha \frac{\sum_{j \in N(i)} a_{ij} \hat{\rho}_j}{\sum_{j \in N(i)} a_{ij}} \quad (4.7)$$

with ρ_i the original measure on the node i , $\hat{\rho}_i$ the smoothed version of the measure, a_{ij} the affinities between nodes i and j , and $N(i)$ the set of neighbors of node i . In a matrix form it becomes

$$\hat{\boldsymbol{\rho}} = \alpha \mathbf{P} \hat{\boldsymbol{\rho}} + (1 - \alpha) \boldsymbol{\rho}, \quad (4.8)$$

and finally

$$\hat{\boldsymbol{\rho}} = (1 - \alpha) (\mathbf{I} - \alpha \mathbf{P})^{-1} \boldsymbol{\rho}, \quad (4.9)$$

with $\mathbf{P}$ being the transition probabilities matrix.

The smoothed strength and the smoothed clustering coefficient are then obtained by replacing ρ by their values on the nodes.

4.1.4 Global density scores

As the methods relying on identifying dense, highly connected subgraphs are often time consuming, and contrarily to the previous measures which only use local information around a node, global density indices take into account the whole structure of the graph, as in [99], which estimates the density on an unweighted k-NN graph from its adjacency matrix. The **Sum-over-Forests (SoF) density index** developed hereunder also computes a density score on nodes using information from the whole graph, by enumerating forests with the matrix forest theorem [24]. This method is based on a sum-over-forests statistical physics framework.

The objective of the next sections is to define the density index, which uses the same Boltzmann probability as in Chapter 3. Before diving into the details, let us briefly describe the main ideas behind the model. In a first step, the set of forests in the graph is enumerated through the matrix-forest theorem and a probability distribution is assigned to each individual forest. In a second step, the expected outdegree of each node in a forest is computed through a sum-over-forests statistical physics formalism, providing a measure of density on the set of nodes.

4.2 The SoF density index

Following Chapter 3, and arguments inspired from [131], the density index can be computed from a quantity appearing in the denominator of the expression of the Boltzmann

probability distribution:

$$P(\varphi) = \frac{\exp[-\theta C(\varphi)]}{\sum_{\varphi' \in \mathcal{F}} \exp[-\theta C(\varphi')]} \quad (4.10)$$

which is

$$\mathcal{Z} = \sum_{\varphi \in \mathcal{F}} \exp[-\theta C(\varphi)] \quad (4.11)$$

where, as a reminder, $C(\varphi)$ is the cost of the forest φ . Equation 4.11 corresponds to the **partition function** in statistical physics (see [71] or any textbook in statistical physics; for instance [129, 140]). For this purpose, let us further define the **free energy** F in the usual way [129, 140] as

$$F = -\frac{1}{\theta} \log(\mathcal{Z}) = -T \log(\mathcal{Z}) \quad (4.12)$$

where $T = 1/\theta$ is the **temperature** of the system. The **expected number of times** a link $k \rightarrow k'$ is present in a forest can easily be computed through (see [129, 140])

$$\bar{\eta}(k, k') = \frac{\partial F}{\partial c_{kk'}} = -\frac{1}{\theta} \frac{\partial(\log \mathcal{Z})}{\partial c_{kk'}} \quad (4.13)$$

$$= \sum_{\varphi \in \mathcal{F}} \frac{\exp[-\theta C(\varphi)]}{\mathcal{Z}} \delta(\varphi; k, k') \quad (4.14)$$

$$= \sum_{\varphi \in \mathcal{F}} P(\varphi) \delta(\varphi; k, k') \quad (4.15)$$

where $\delta(\varphi; k, k')$ is a Kronecker delta indicating if the link $k \rightarrow k'$ is present in forest φ . The **expected outdegree** of node k on a forest, which defines the **SoF density index**, is

$$\text{dens}(k) = \sum_{\varphi \in \mathcal{F}} P(\varphi) \left(\sum_{k'=1}^n \delta(\varphi; k, k') \right) = \sum_{k'=1}^n \bar{\eta}(k, k') \quad (4.16)$$

and corresponds to the sum of the contributions of the arcs issued from node k .

In the next section, we show that the partition function can easily be computed from the cost matrix.

4.3 Computation of the partition function $\mathcal{Z}$

By using the matrix-forest theorem [23, 24], let us show how the partition function $\mathcal{Z}$ (Equation (4.11)) can be computed exactly from the immediate costs. Indeed, consider a graph G characterized by an adjacency matrix $\mathbf{A}$ containing the weights on the arcs. The matrix-forest theorem states that (see [24], lemma 2, or [23] for details):

$$\det(\mathbf{I} + \mathbf{L}(\mathbf{A})) = \varepsilon(\mathcal{F}), \quad (4.17)$$

where $\varepsilon(\mathcal{F})$ is the *sum* of the total weights of all the rooted (diverging in the directed case) forests $\varphi \in \mathcal{F}$ that can be extracted from the graph. The total weight of a particular rooted forest φ is the *product* of the weights of the individual arcs defining it.

We now apply this concept to the matrix $\mathbf{W}$, already used in Chapter 3, and defined from the cost matrix, $\mathbf{C}$,

$$\mathbf{W} = \exp[-\theta \mathbf{C}], \quad (4.18)$$

where the exponential function is taken elementwise. Thus, the elements of matrix $\mathbf{W}$ are $\exp[-\theta c_{kk'}]$. Now, if we set as adjacency matrix $\mathbf{A} = \mathbf{W}$, the total weight of a rooted forest φ corresponds to the product of the individual weights defining it, i.e., $\prod_{k,k':k \rightarrow k' \in \varphi} a_{kk'} = \prod_{k,k':k \rightarrow k' \in \varphi} \exp[-\theta c_{kk'}] = \exp[-\theta \sum_{k,k':k \rightarrow k' \in \varphi} c_{kk'}] = \exp[-\theta C(\varphi)]$. We can immediately deduce from the matrix-forest theorem that $\det(\mathbf{I} + \mathbf{L}(\mathbf{W}))$, where $\mathbf{L}(\mathbf{W}) = \mathbf{Diag}(\mathbf{W}^T \mathbf{e}) - \mathbf{W}$, is equal to $\sum_{\varphi \in \mathcal{F}} \exp[-\theta C(\varphi)] = \mathcal{Z}$. Therefore,

$$\mathcal{Z} = \det(\mathbf{I} + \mathbf{L}(\mathbf{W})), \text{ with } \mathbf{W} = \exp[-\theta \mathbf{C}]. \quad (4.19)$$

This result is used in the next section in order to derive the SoF density index.

Notice that, interestingly, the transform (4.18) coincides, up to a multiplier, with transformation (18) in [21] and (27) in [22].

4.4 Computation of the SoF density index

Now that we have seen how to compute the partition function $\mathcal{Z}$, we turn to the computation of the density index that can be deduced from $\mathcal{Z}$ thanks to Equations (4.13) and (4.16).

We thus have to compute the derivatives of $\mathcal{Z}$ (Equation (4.19)) in terms of $c_{kk'}$ (see Equation (4.13)) in order to obtain the different quantities of interest. Now, it is well-known (see, e.g., [61, 139]) that $\partial \log(\det(\mathbf{X}))/\partial t = \text{trace}(\mathbf{X}^{-1} \frac{\partial \mathbf{X}}{\partial t})$. Thus, for the expected number of times the link $k \rightarrow k'$ appears in a forest, we obtain

$$\begin{aligned} \bar{\eta}(k, k') &= \frac{\partial F}{\partial c_{kk'}} = -\frac{1}{\theta} \frac{\partial \log(\det(\mathbf{I} + \mathbf{L}(\mathbf{W})))}{\partial c_{kk'}} \\ &= -\frac{1}{\theta} \text{trace}(\mathbf{Z} \frac{\partial(\mathbf{I} + \mathbf{L}(\mathbf{W}))}{\partial c_{kk'}}) \\ &= -\frac{1}{\theta} \text{trace}(\mathbf{Z} \frac{\partial \mathbf{L}(\mathbf{W})}{\partial c_{kk'}}) \\ &= -\frac{1}{\theta} \text{trace}(\mathbf{Z} \frac{\partial(\mathbf{D} - \mathbf{W})}{\partial c_{kk'}}), \end{aligned} \quad (4.20)$$

where the matrix $\mathbf{Z}$ is defined as

$$\mathbf{Z} = (\mathbf{I} + \mathbf{L}(\mathbf{W}))^{-1} = (\mathbf{I} + (\mathbf{Diag}(\mathbf{W}^T \mathbf{e}) - \mathbf{W}))^{-1}, \quad (4.21)$$

and is equal to the $\mathbf{K}_{\text{BoF}}$ kernel defined in Chapter 3.

Now, we easily find that $\partial \mathbf{W} / \partial c_{kk'} = -\theta w_{kk'} \mathbf{e}_k \mathbf{e}_{k'}^T$ and $\partial \mathbf{D} / \partial c_{kk'} = -\theta w_{kk'} \mathbf{e}_{k'} \mathbf{e}_k^T$ so that

$$\frac{\partial \mathbf{L}(\mathbf{W})}{\partial c_{kk'}} = \frac{\partial (\mathbf{D} - \mathbf{W})}{\partial c_{kk'}} = -\theta w_{kk'} (\mathbf{e}_{k'} \mathbf{e}_k^T - \mathbf{e}_k \mathbf{e}_{k'}^T), \quad (4.22)$$

where $\mathbf{e}_k$ is a basis column vector with zeroes everywhere except in position k where there is a 1.

Thus, by defining $\mathbf{z}_k = \text{col}_k(\mathbf{Z})$ as column k of matrix $\mathbf{Z}$,

$$\begin{aligned} \bar{\eta}(k, k') &= \text{trace}(w_{kk'} \mathbf{Z} (\mathbf{e}_{k'} \mathbf{e}_k^T - \mathbf{e}_k \mathbf{e}_{k'}^T)) \\ &= w_{kk'} (\text{trace}(\mathbf{z}_{k'} \mathbf{e}_k^T) - \text{trace}(\mathbf{z}_k \mathbf{e}_{k'}^T)) \\ &= w_{kk'} z_{k'k} - w_{kk'} z_{k'k}. \end{aligned} \quad (4.23)$$

Therefore, the expected outdegree of node k – the **SoF density index** of node k – is

$$\text{dens}(k) = \sum_{k'=1}^n \bar{\eta}(k, k') = \sum_{k'=1}^n (w_{kk'} z_{k'k} - w_{kk'} z_{k'k}), \quad (4.24)$$

where we used Equations (4.16) and (4.23). The $n \times 1$ column vector containing the elements $\text{dens}(k)$ will be called $\mathbf{d}$, with

$$\mathbf{d} = \mathbf{W} \text{diag}(\mathbf{Z}) - \text{diag}(\mathbf{WZ}), \quad (4.25)$$

and where $\text{diag}(\mathbf{X})$ is a column vector containing the diagonal of matrix $\mathbf{X}$.

The SoF index can therefore be found by applying the following, simple, procedure:

1. Compute the $\mathbf{W}$ matrix using Equation (4.18).
2. Compute the $\mathbf{Z}$ matrix using Equation (4.21).
3. Compute the column vector $\mathbf{d}$ containing the SoF index of each node using Equation (4.25).

As the algorithm computing the SoF density index requires the inversion of the matrix $(\mathbf{I} + \mathbf{L}(\mathbf{W}))^{-1}$ which has a complexity of $\mathcal{O}(n^3)$, matrix-vector multiplication, which has a complexity of $\mathcal{O}(n^2)$, and matrix-matrix multiplication, having a $\mathcal{O}(n^3)$ complexity [27], its overall time complexity is $\mathcal{O}(n^3)$.

4.5 Experiments

In this experimental section, the SoF density index is assessed on the identification of dense regions on graphs. Unlike classical clustering methods, the goal here is not to find an exact partition of the data, but only regions of graphs where the nodes are tightly aggregated, suggesting some community-like structure.

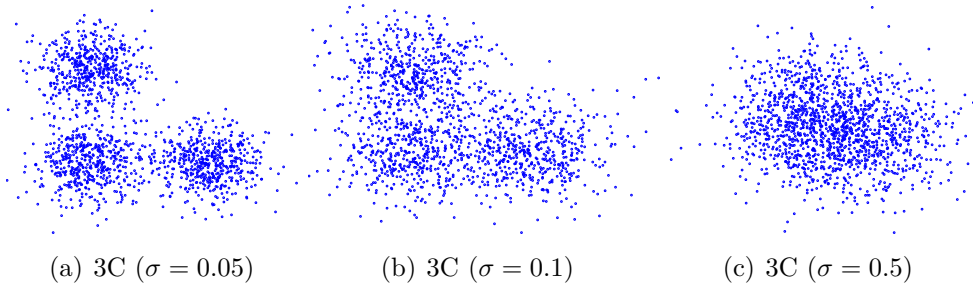


Figure 4.1: 3-communities (3C) datasets.

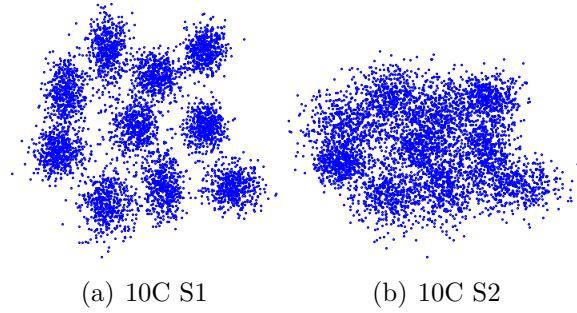


Figure 4.2: 10-communities (10C) datasets.

4.5.1 Datasets

The performance of the SoF density index is assessed on 16 datasets belonging to five groups: 3-communities, 10-communities, S-Sets, real datasets and Grid dataset.

The **3-communities** (resp. **10-communities**) datasets are artificial datasets we built: each one is made of three (resp. ten) clusters, created using Gaussian distributions $N(\mu, \sigma)$, μ being the mean (the center of the cluster) and σ^2 the variance of the data. Each cluster is made of 500 nodes, lying in two dimensions. Three values of σ (illustrating various degree of overlapping between the communities) were used to build graphs in the 3-communities case: 0.05, 0.1, 0.5 (the standard deviation is the same in each direction, giving isotropic communities). For the 10-communities datasets, the σ values are different in the two space directions, (x, y) . These values, called σ_x and σ_y are reported in Table 4.1 for two sets : S_1 with small overlapping and S_2 with strong overlapping. The 3/10-communities datasets are illustrated in Figures 4.1 and 4.2.

The **S-Sets** [53] include two datasets: S2 and S4. They are also based on artificial data and are composed of 5000 two-dimensional observations each, grouped in 15 clusters of various shapes. Figure 4.3 illustrates S2, with well separated clusters and S4, showing more overlapped ones.

Real datasets (coming from real experiments) are also used. They are generated from

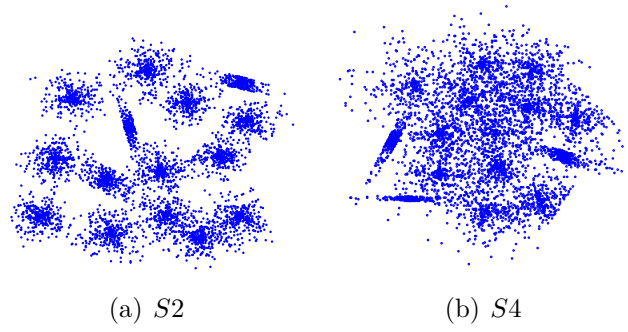


Figure 4.3: S-Sets datasets.

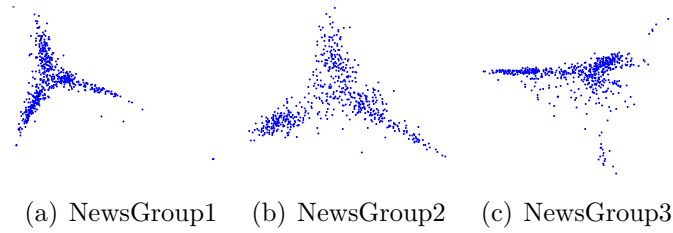


Figure 4.4: NewsGroup datasets.

the **NewsGroup** dataset (see Figure 4.4): this dataset is originally composed of about 20,000 unstructured documents, taken from 20 discussion groups (newsgroups) of the Usenet diffusion list, and composed of 20 classes. For our experiments, three subsets related to different topics are extracted from the original database (NewsGroup1, 2, and 3) [173]. The graphs of documents were built by sampling at random about 200 documents in each of three classes from three different topics.

Finally, the **Grid** dataset (shown in Figure 4.5) is composed of two rectangular subgrids (of respectively 100 and 400 points), each one with a different density (i.e., the points are farther from each other in one subgrid than in the other).

S1	σ_x	0.8	0.5	0.5	0.8	1	1	0.5	0.5	0.5	1
	σ_y	0.8	0.5	0.5	0.5	1	0.5	1	1	1	0.5
S2	σ_x	1.8	1.5	1	1.8	2	1	2.5	1.5	1	3
	σ_y	1.6	1	2.5	3	2	1	2	2	3	1.5

Table 4.1: (σ_x, σ_y) (standard deviations) values for the 10-communities datasets, for two degrees of overlapping between the clusters (S1 small overlapping, S2 strong overlapping).

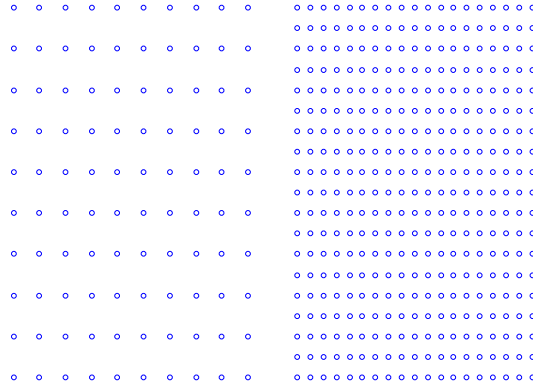


Figure 4.5: Grid dataset.

4.5.2 Graph construction

We constructed the graphs corresponding to the 3/10-communities, the S-Sets datasets and the Grid dataset using two classical methods: the ϵ -graph and the k-nearest neighbors (k-NN).

The **ϵ -graph** computes the Euclidean distance between each pair of observations in the dataset and transforms it into an affinity using

$$a_{ij} = \exp \left[-\frac{d_{ij}^2}{\sigma^2} \right] \quad (4.26)$$

where d_{ij} is the Euclidean distance between nodes i and j , and σ^2 is the variance of the distances between all the observations in the dataset. This is a standard way for computing a similarity graph from distances between data, frequently used in spectral clustering (see, e.g., [166, 169] – up to a scaling factor of the sigma parameter, depending on the authors). The nodes are then linked to others only if they show an affinity superior to a certain threshold (80, 90, 95, and 99 percentiles were used). The resulting graphs are undirected, and both the weighted case (where edges bear the nodes affinities) and the unweighted case are investigated.

The **k-NN** graph construction method simply links a node to its k nearest neighbors, i.e., those who have the highest affinity with that node. This relation is not symmetric, giving birth to directed graphs. We transform them into undirected graphs using

$$\mathbf{A} \leftarrow \max(\mathbf{A}, \mathbf{A}^T) \quad (4.27)$$

where $\mathbf{A}$ is the adjacency matrix of the created graph, and the maximum operator is taken elementwise.

For the NewsGroup datasets, the graphs were already built [173] and only the adjacency matrices are at our disposal. To visualize those graphs, we use the diffusion maps

embedding method [84, 112, 175] in two dimensions, whose output is the new spatial coordinates of the nodes. The corresponding graphs are reconstructed with the ϵ -graph method, allowing us to compute the density index on the nodes. Indeed, trying to proceed inversely (computing the densities before the diffusion map embedding) is not visually accurate: during the embedding, the nodes are spatially rearranged and the color of the nodes (indicating high or low density, see the visual results below) does not reflect the true density of the 2-D embedding.

The cost matrices used in the evaluation of the SoF density index are then computed as the reciprocals of the affinity matrices constructed above.

4.5.3 Evaluation methods

We use three methods to evaluate to which extent the high density areas are well identified: Spearman’s correlation, visual checking, and a k-Core procedure. The correlation and the k-Core are only applicable to datasets whose “true” density is known.

Firstly, since the probability density function is known for every node of the 3/10-communities datasets (i.e., the exact parameters’ values of the Gaussian distributions are known), we compute **Spearman’s correlation** (Pearson’s linear correlation and Kendall’s correlation show similar trends) between those “true” densities and the SoF densities.

Secondly, we perform a **visual checking** on the graphs by superimposing the density index on the representation of the nodes. This is done by assigning each node a color: from dark blue for nodes presenting a low density value to dark red for nodes presenting a high density value.

Thirdly, we use a generalized k-Core procedure with our Sum-over-Forests density index in input, in order to validate this index as an estimator of density on a graph. Indeed, for artificial datasets on which the true density is known, we control the density score on each node. By using a generalized k -core strategy (removing nodes having a density score inferior to a certain threshold), the algorithm gradually converges to dense areas on the graph. We then compare density measures – like the strength and the SoF density score – to the true density, and see if they converge to the same dense regions than the true density. During the experiments, we observed that updating the SoF density at each iteration did not provide better results than computing the density once at the beginning, and removing the node presenting the lowest density score. This can be explained by the good fitting of the SoF density index to the true density. This also have the effect to greatly reduce the computational load of this “non-iterative generalized k -core”. In order to evaluate to which extent the high density areas are well identified

with the generalized k -cores, we first draw curves representing the evolution of the true density's mean value on the graph, while removing the node having the lowest measure. The closer a curve is to the true density curve, the more likely the corresponding measure identifies well the dense zones on the graph. In order to have a quantifiable measure of how close is a density measure to the true density, we also compute an "area under curve" (AUC) score, defined as the area below the curve of the corresponding measure, compared to the area below the true density curve. A score of one indicates that the measure identifies perfectly dense zones. The results obtained with the SoF density index are compared with another measure used as a baseline: the strength.

We can then assess the quality of the SoF as a density estimator, and use it as a basis for a density-based clustering algorithm which is described in the next chapter.

Concerning the tuning of the θ parameter in the SoF method, we used the correlation method on the 3/10-communities graphs. The parameter's value giving the highest correlation score (for threshold graphs, $\theta = 10$ and for k-NN graphs, $\theta = 100$) is then used for the 3/10-communities as well as for the S-Sets and Newsgroup datasets.

The results obtained with the SoF density index are finally compared with two other measures for identifying dense zones: the strength (Str) and the clustering coefficient (CC).

4.5.4 Results

4.5.4.1 Correlation results

The correlation results for 3/10-communities showing a comparison between SoF density index, the strength and the clustering coefficient are displayed in Figure 4.6 and 4.7, and in Tables 4.2 to 4.6, where the results for a set of θ values can be seen.

When using the k-NN for constructing graphs, the SoF density index is clearly superior to the strength and to the clustering coefficient (the latter performs badly in every situation). This may be explained by the fact that the information concerning the connectivity is useless in this case, as all the nodes have theoretically almost the same degree. The SoF density index then makes a better use of the affinities borne by the arcs of the graphs than the strength does. This superiority may have two causes:

- **Smoothing:** The value of the parameter θ acts as a smoothing parameter of the density index on the nodes of the graph. Indeed, low values of θ favor high-cost forests (which can be seen as forests composed of large trees) in the Boltzmann distribution, and thus involve a larger part of the graph to compute the SoF density

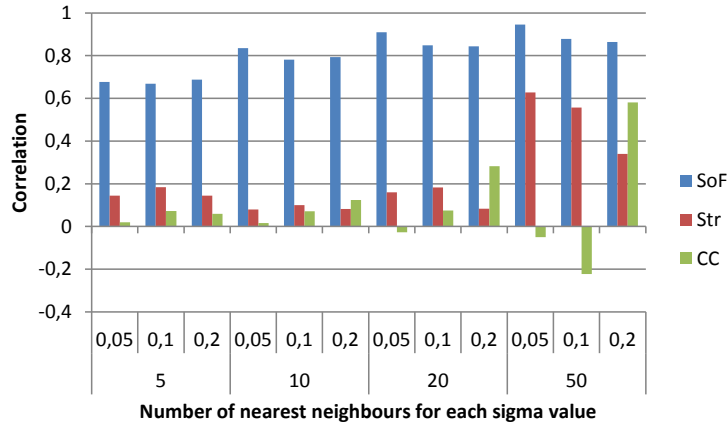
index. The information used is then no more local, as with the strength, but is globalized.

- **Sampling:** As we sample forests on the graph, and take the expected value of the outdegree of a node, averaged over all possible forests, situations where local affinities do not bring much information are averaged out, and thus attenuated.

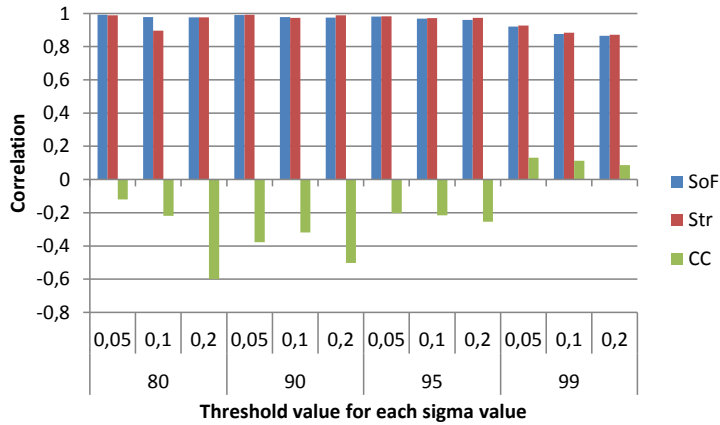
When using the ϵ -graph construction method, the results are not so clear. The results obtained by the strength and the SoF index for the 3-communities case have a correlation with the true density of almost 1 and are practically identical (only the weighted case is represented here in Figure 4.6(b), as the unweighted case gives similar results). The SoF index is clearly better on the 10-communities datasets in the weighted case and for low threshold (ϵ) values. Indeed, the number of neighbors increases dramatically when the threshold is low, and again the information of connectivity becomes useless, nodes having each a large degree value. The only useful information are the affinities and, as before, the SoF density index uses it in a more efficient way than the strength. When the threshold is higher the two measures converge to the same value. In the unweighted case (no affinity information), the SoF index and the strength behave similarly: the correlations are low for low threshold values and increase when the threshold increases. Notice that in Figure 4.7(c), we somewhat surprisingly observe a negative correlation between the SoF density/the strength and the true density, which can be explained by the fact that, due to finite sampling conditions, areas with low density are sometimes wrongly identified as high density regions (see Figure 4.8). The good results of both the SoF index and the strength in the 3-communities dataset with threshold and unweighted arcs are probably due to the fact that these datasets are quite smaller and simpler to handle, having only 3 clusters distributed with Gaussians of the same variance in all directions.

The smoothing versions of the strength and the clustering coefficient are also evaluated using the correlation procedure. We report in Tables 4.7 and 4.8 the correlation results between the smoothed version of the strength using the Laplacian regularization and the Page-Rank smoothing, and the “true” density on the 10C-S2 dataset, which may be considered as the “hardest” one, among the datasets for which the ground-truth density is known. We can see that the smoothed versions of the strength do not surpass the original strength, except for a few times. The same conclusion can be made for Tables 4.9 and 4.10, where the correlation results for the smoothed versions of the clustering coefficient are shown. Hence, we will not use those smoothed density indices in the rest of this chapter.

A first conclusion can be drawn so far: the SoF density index is much more stable and independent from the type of graph than the strength and the clustering coefficient.

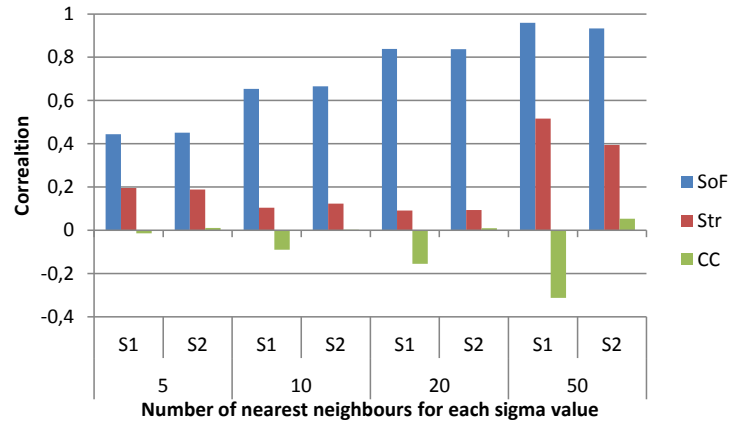


(a) 3-communities - k-NN

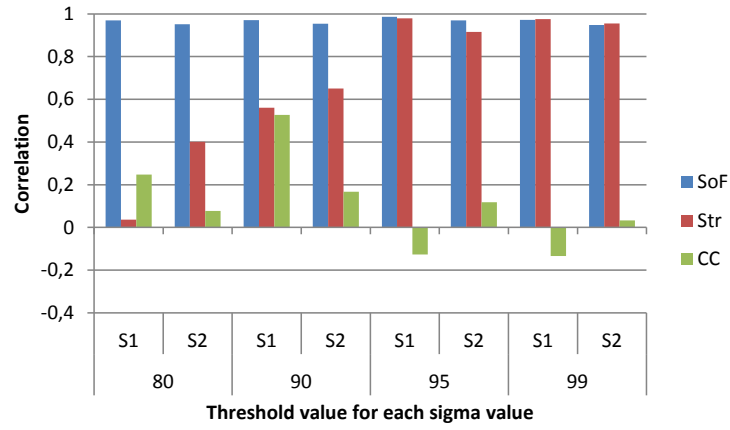


(b) 3-communities - Th - W

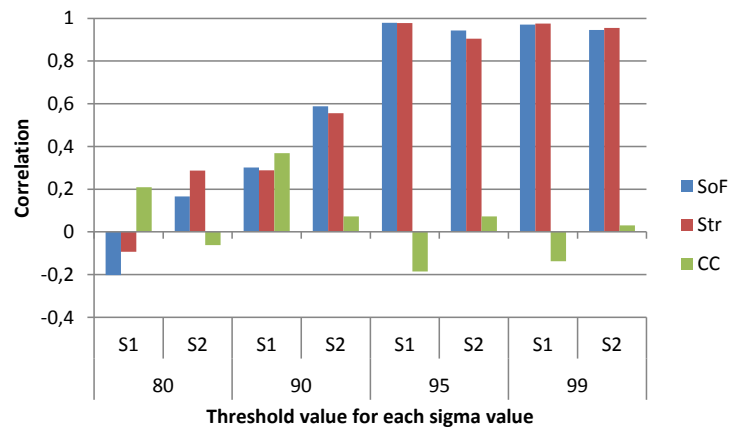
Figure 4.6: Correlation results for the 3-communities datasets, between true density and SoF density index, Strength, CC. Weighted (W) and Unweighted (U) graphs are considered, constructed with k-Nearest Neighbors (k-NN) or Threshold (Th) methods.



(a) 10-communities - k-NN



(b) 10-communities - Th - W



(c) 10-communities - Th - U

Figure 4.7: Correlation results for the 10-communities datasets, between true density and SoF density index, Strength, CC. Weighted (W) and Unweighted (U) graphs are considered, constructed with k-Nearest Neighbors (k-NN) or Threshold (Th) methods.

3C $\sigma = 0.05$		Str			SoF		
			0.01	0.1	1	10	100
Weighted	th80	0,9881	0.9910	0.9915	0.9939	0.9851	0.9106
Weighted	th90	0,9914	0.9696	0.9703	0.9774	0.9848	0.9106
Weighted	th95	0,9822	0.8309	0.8360	0.8816	0.9801	0.9106
Weighted	th99	0,9261	0.5297	0.5438	0.6951	0.9260	0.9001
Binary	th80	0,9779	0.9810	0.9811	0.9815	0.9782	0.9779
Binary	th90	0,9908	0.9616	0.9619	0.9657	0.9908	0.9908
Binary	th95	0,982	0.8182	0.8227	0.8647	0.9816	0.9820
Binary	th99	0,9258	0.5285	0.5427	0.6905	0.9187	0.9258
Weighted	NN5	0,1448	0.0537	0.0533	0.0593	0.3704	0.7569
Weighted	NN10	0.0794	0.1505	0.1492	0.1464	0.5683	0.8561
Weighted	NN20	0,1601	0.3037	0.3013	0.2918	0.7742	0.8968
Weighted	NN50	0,6281	0.6071	0.6051	0.5986	0.9519	0.9102

Table 4.2: Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.05$.

3C $\sigma = 0.1$		Str			SoF		
			0.01	0.1	1	10	100
Weighted	th80	0,8958	0.9825	0.9825	0.9773	0.9661	0.8055
Weighted	th90	0,9734	0.9280	0.9296	0.9440	0.9660	0.8055
Weighted	th95	0,9717	0.7559	0.7629	0.8303	0.9635	0.8055
Weighted	th99	0,8829	0.4579	0.4710	0.6206	0.8818	0.8020
Binary	th80	0,8173	0.9665	0.9666	0.9674	0.8190	0.8173
Binary	th90	0,9689	0.9119	0.9126	0.9217	0.9694	0.9689
Binary	th95	0,9709	0.7381	0.7439	0.8026	0.9700	0.9709
Binary	th99	0,8828	0.4562	0.4688	0.6130	0.8720	0.8828
Weighted	NN5	0,1838	0.0672	0.0674	0.0817	0.4334	0.6976
Weighted	NN10	0.1004	0.1638	0.1611	0.1577	0.6131	0.7688
Weighted	NN20	0,1828	0.3360	0.3320	0.3238	0.7812	0.8001
Weighted	NN50	0,5575	0.5995	0.5964	0.6004	0.9350	0.8055

Table 4.3: Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.1$.

3C $\sigma = 0.5$		Str			SoF			
			0.01	0.1	1	10	100	
Weighted	th80	0,9957	0.9688	0.9692	0.9499	0.9712	0.8351	
Weighted	th90	0,991	0.8083	0.8119	0.8449	0.9711	0.8351	
Weighted	th95	0,9801	0.5480	0.5602	0.6891	0.9690	0.8351	
Weighted	th99	0,9144	0.4927	0.5062	0.6577	0.9134	0.8334	
Binary	th80	0,9955	0.9482	0.9484	0.9518	0.9955	0.9955	
Binary	th90	0,9911	0.7839	0.7880	0.8259	0.9910	0.9911	
Binary	th95	0,9801	0.5271	0.5369	0.6467	0.9793	0.9801	
Binary	th99	0,9149	0.4917	0.5046	0.6504	0.9061	0.9149	
Weighted	NN5	0,1662	0.1028	0.1031	0.1299	0.5078	0.7477	
Weighted	NN10	0.1053	0.2400	0.2356	0.2312	0.6776	0.8077	
Weighted	NN20	0,142	0.4746	0.4620	0.4170	0.8491	0.8317	
Weighted	NN50	0,3925	0.7948	0.7718	0.6789	0.9445	0.8350	

Table 4.4: Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 3-communities dataset with $\sigma = 0.5$.

10C S1		Str			SoF		
			0.01	0.1	1	10	100
Weighted	th80	0,0366	0.0656	0.1233	0.7252	0.9842	0.9693
Weighted	th90	0,5607	0.6256	0.6558	0.8479	0.9842	0.9693
Weighted	th95	0,9794	0.9488	0.9492	0.9532	0.9867	0.9693
Weighted	th99	0,9756	0.6842	0.6930	0.7901	0.9756	0.9662
Binary	th80	-0.0935	-0.2116	-0.2116	-0.2115	-0.0987	-0.0935
Binary	th90	0,2882	0.2646	0.2647	0.2664	0.2895	0.2882
Binary	th95	0,9778	0.9442	0.9443	0.9463	0.9782	0.9778
Binary	th99	0.9755	0.6819	0.6905	0.7855	0.9748	0.9755
Weighted	NN5	0,1959	0.0276	0.0273	0.0259	0.2287	0.5964
Weighted	NN10	0,1046	0.0651	0.0647	0.0603	0.2655	0.7812
Weighted	NN20	0,091	0.1634	0.1634	0.1637	0.4595	0.9001
Weighted	NN50	0,5165	0.4556	0.4566	0.4663	0.8355	0.9590

Table 4.5: Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 10C-S1 dataset.

10C S2		Str		SoF				
				0.01	0.1	1	10	100
Weighted	th80	0,4009	0.4153	0.4573	0.7661	0.9640	0.9111	
Weighted	th90	0,6509	0.7268	0.7411	0.8449	0.9640	0.9111	
Weighted	th95	0,9157	0.8715	0.8728	0.8881	0.9671	0.9111	
Weighted	th99	0,9549	0.5434	0.5522	0.6613	0.9549	0.9102	
Binary	th80	0,2873	0.1308	0.1308	0.1313	0.2834	0.2873	
Binary	th90	0,5553	0.5531	0.5532	0.5551	0.5562	0.5553	
Binary	th95	0,9049	0.8571	0.8576	0.8655	0.9059	0.9049	
Binary	th99	0,9546	0.5392	0.5476	0.6514	0.9528	0.9546	
Weighted	NN5	0,1889	0.0328	0.0328	0.0355	0.2596	0.5678	
Weighted	NN10	0,1229	0.0588	0.0585	0.0584	0.3519	0.7488	
Weighted	NN20	0,0933	0.1320	0.1311	0.1286	0.5723	0.8565	
Weighted	NN50	0,3949	0.3908	0.3901	0.3945	0.8979	0.9066	

Table 4.6: Spearman’s correlations between the Strength (Str), the SoF density index (SoF), and the “true density” on 10C-S2 dataset.

10C S2		Str		StrLap					
				0,01	0,1	0,3	0,5	0,7	0,9
Weighted	th80	0.4009	0.4063	0.4041	0.4024	0.4019	0.4017	0.4015	0.4015
Weighted	th90	0.6509	0.5499	0.4981	0.4859	0.4828	0.4815	0.4807	0.4805
Weighted	th95	0.9157	0.8009	0.6303	0.5811	0.5644	0.5557	0.5500	0.5481
Weighted	th99	0.9549	0.9601	0.9268	0.8471	0.7931	0.7558	0.7282	0.7181
Binary	th80	0.2873	0.3455	0.3496	0.3499	0.3499	0.3499	0.3499	0.3499
Binary	th90	0.5553	0.4940	0.4630	0.4536	0.4513	0.4503	0.4497	0.4495
Binary	th95	0.9049	0.7788	0.6195	0.5748	0.5596	0.5518	0.5469	0.5452
Binary	th99	0.9546	0.9601	0.9252	0.8439	0.7896	0.7523	0.7248	0.7148
Weighted	NN5	0.1889	0.0606	-0.0304	-0.0534	-0.0685	-0.0800	-0.0891	-0.0927
Weighted	NN10	0.1229	0.0494	-0.0106	-0.0172	-0.0216	-0.0254	-0.0292	-0.0307
Weighted	NN20	0.0933	0.0620	0.0704	0.0709	0.0575	0.0443	0.0332	0.0290
Weighted	NN50	0.3957	0.4319	0.4418	0.3327	0.2656	0.2234	0.1952	0.1853

Table 4.7: Spearman’s correlations between the Strength (Str), the smoothed version of Str with Laplacian smoothing (StrLap), and the “true density” on 10C-S2 dataset.

10C S2		Str	StrPR						
			0,01	0,1	0,3	0,5	0,7	0,9	0,99
Weighted	th80	0.4009	0.4010	0.4014	0.4018	0.4010	0.3977	0.3853	0.3609
Weighted	th90	0.6509	0.6502	0.6434	0.6258	0.6035	0.5718	0.5192	0.4679
Weighted	th95	0.9157	0.9151	0.9085	0.8875	0.8522	0.7868	0.6433	0.4635
Weighted	th99	0.9549	0.9555	0.9569	0.9599	0.9587	0.9448	0.8522	0.5309
Binary	th80	0.2873	0.2880	0.2940	0.3063	0.3157	0.3212	0.3161	0.3032
Binary	th90	0.5553	0.5550	0.5513	0.5416	0.5283	0.5087	0.4752	0.4338
Binary	th95	0.9049	0.9038	0.8973	0.8757	0.8404	0.7752	0.6384	0.4675
Binary	th99	0.9546	0.9562	0.9568	0.9598	0.9586	0.9446	0.8513	0.5300
Weighted	NN5	0.1889	0.1532	0.0225	-0.0273	-0.0469	-0.0738	-0.1295	-0.1977
Weighted	NN10	0.1229	0.1137	0.0488	0.0011	-0.0123	-0.0210	-0.0428	-0.0990
Weighted	NN20	0.0933	0.0923	0.0751	0.0535	0.0568	0.0600	0.0311	-0.0582
Weighted	NN50	0.3949	0.3957	0.4039	0.4226	0.4370	0.4321	0.3212	0.0705

Table 4.8: Spearman’s correlations between the Strength (Str), the smoothed version of Str with Page-Rank smoothing (StrPR), and the “true density” on 10C-S2 dataset.

10C S2		CC	CCLap						
			0,01	0,1	0,3	0,5	0,7	0,9	0,99
Weighted	th80	0.0777	-0.0936	-0.1203	-0.1224	-0.1228	-0.1230	-0.1231	-0.1231
Weighted	th90	0.1676	-0.0305	-0.1206	-0.1299	-0.1319	-0.1328	-0.1333	-0.1335
Weighted	th95	0.1182	0.0406	-0.1063	-0.1316	-0.1374	-0.1401	-0.1417	-0.1423
Weighted	th99	0.0331	0.0695	0.1754	0.1857	0.1673	0.1502	0.1351	0.1295
Binary	th80	-0.0621	-0.2020	-0.2129	-0.2138	-0.2141	-0.2142	-0.2142	-0.2142
Binary	th90	0.0724	-0.1100	-0.1703	-0.1761	-0.1774	-0.1779	-0.1783	-0.1784
Binary	th95	0.0728	-0.0039	-0.1328	-0.1514	-0.1558	-0.1579	-0.1590	-0.1594
Binary	th99	0.0305	0.0674	0.1728	0.1804	0.1609	0.1434	0.1282	0.1227
Weighted	NN5	0.0113	-0.0201	-0.0434	-0.0603	-0.0719	-0.0804	-0.0872	-0.0897
Weighted	NN10	0.0019	-0.0022	-0.0054	-0.0109	-0.0132	-0.0144	-0.0150	-0.0152
Weighted	NN20	0.0094	0.0117	0.0270	0.0461	0.0556	0.0602	0.0622	0.0625
Weighted	NN50	0.0532	0.0926	0.2336	0.2443	0.2273	0.2123	0.1993	0.1940

Table 4.9: Spearman’s correlations between the clustering coefficient (CC), the smoothed version of CC with Laplacian smoothing (CCLap), and the “true density” on 10C-S2 dataset.

10C S2		CC	CCPR						
			0,01	0,1	0,3	0,5	0,7	0,9	0,99
Weighted	th80	0.0777	0.0764	0.0642	0.0352	0.0055	-0.0268	-0.0605	-0.0720
Weighted	th90	0.1676	0.1665	0.1560	0.1275	0.0901	0.0420	-0.0214	-0.0593
Weighted	th95	0.1182	0.1179	0.1146	0.1042	0.0866	0.0557	-0.0058	-0.0524
Weighted	th99	0.0331	0.0337	0.0393	0.0567	0.0848	0.1277	0.1962	0.1714
Binary	th80	-0.0621	-0.0631	-0.0722	-0.0930	-0.1144	-0.1367	-0.1545	-0.1590
Binary	th90	0.0724	0.0715	0.0627	0.0395	0.0092	-0.0289	-0.0733	-0.0930
Binary	th95	0.0728	0.0726	0.0700	0.0613	0.0462	0.0195	-0.0325	-0.0657
Binary	th99	0.0305	0.0310	0.0366	0.0538	0.0816	0.1242	0.1921	0.1679
Weighted	NN5	0.0113	0.0001	-0.0266	-0.0388	-0.0487	-0.0633	-0.0961	-0.1575
Weighted	NN10	0.0019	0.0015	-0.0023	-0.0038	-0.0049	-0.0082	-0.0124	-0.0521
Weighted	NN20	0.0094	0.0095	0.0105	0.0135	0.0181	0.0276	0.0553	0.0518
Weighted	NN50	0.0532	0.0539	0.0612	0.0828	0.1167	0.1715	0.2424	0.1660

Table 4.10: Spearman’s correlations between the clustering coefficient (CC), the smoothed version of CC with Page-Rank smoothing (CCPR), and the “true density” on 10C-S2 dataset.

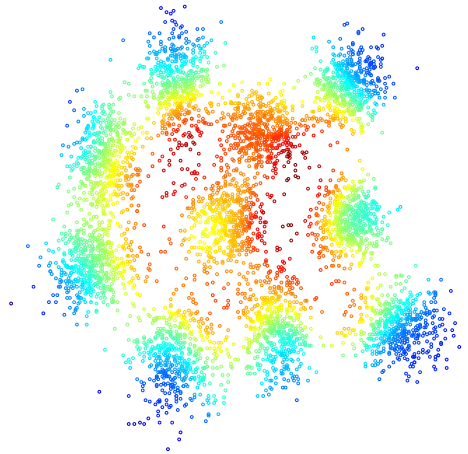


Figure 4.8: 10-communities dataset (low sigma values S1), with SoF superimposed (Threshold 80, Unweighted) and having a negative correlation with the “true” density.

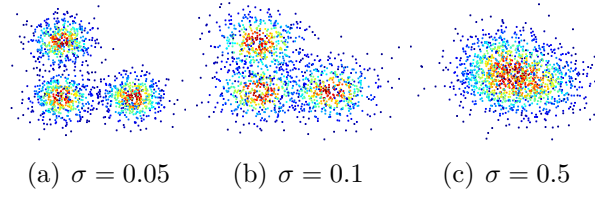


Figure 4.9: 3-communities datasets for various σ values with the true density superimposed.

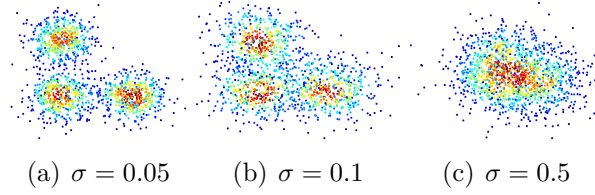


Figure 4.10: 3-communities datasets for various σ values with the SoF density index superimposed.

4.5.4.2 Visual results

The visual results confirm the correlation results described above. As there are many different cases for the 3/10-communities datasets, only few visual examples, representative of the overall behavior of the density measures, are shown. For instance, Figure 4.10 shows the 3-communities datasets (threshold 95, weighted) with the SoF density index superimposed, while the Figure 4.9 displays the “true” density. It can be observed on this simple example that the SoF density index is able to recover the dense areas of the clusters. In the 10-communities datasets, a visual confirmation is given in Figures 4.11, 4.12, 4.13 and 4.14. The SoF density index is visually very close to the true density and the highly dense regions are well identified. The sensitivity of the strength to the way a graph is constructed can also be seen on these figures. Indeed, going from graphs constructed with the ϵ -graph method with threshold 95, and going to threshold 90, the strength merges the dense areas which were quite well defined with a 95 threshold. This is not the case with the SoF density index, which stays much more stable regarding the way a graph is constructed.

The dense regions on the S-Sets are also well identified. Figure 4.15 shows that the SoF density index is able to recover the 15 densest areas on the S2 and S4 graphs, even if they are tightly aggregated (the center of the dense regions are well separated). The strength and the clustering coefficient are illustrated on Figures 4.16 and 4.17, showing poor results, mainly on S4. The Newsgroup datasets confirm the results obtained so far (Figure 4.18). For SoF density index, the three clusters are recovered on those graphs, except on NewsGroup3 where two clusters are too tightly intricated to be differentiated.

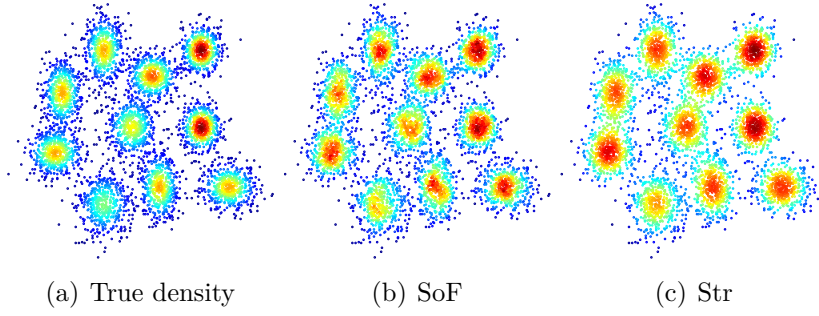


Figure 4.11: 10-communities dataset (low sigma values S1) with true density, SoF density index, and Strength superimposed (Threshold 95, Weighted).

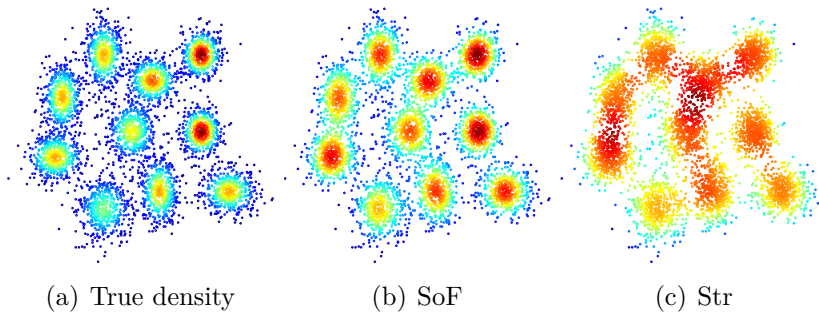


Figure 4.12: 10-communities dataset (low sigma values S1) with true density, SoF density index, and Strength superimposed (Threshold 90, Weighted).

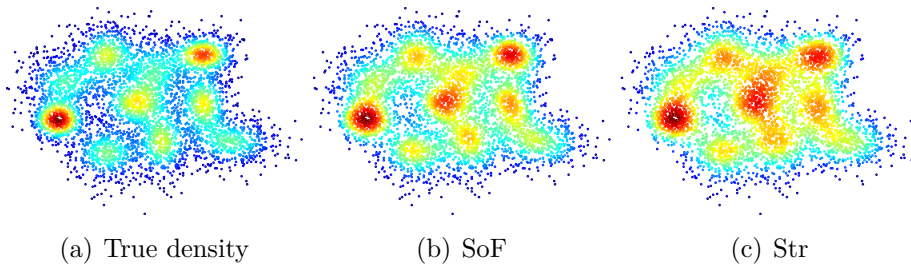


Figure 4.13: 10-communities dataset (high sigma values S2) with true density, SoF density index, and Strength superimposed (Threshold 95, Weighted).

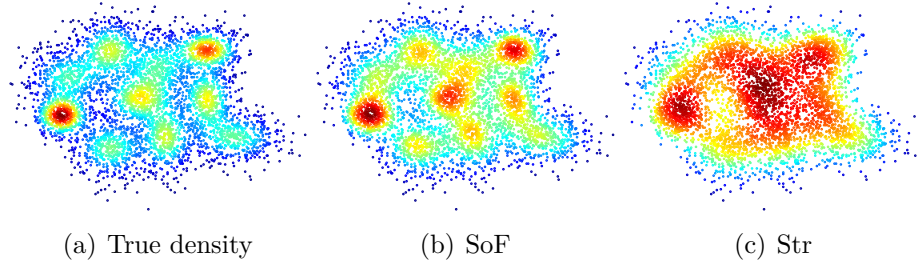


Figure 4.14: 10-communities dataset (high sigma values S2) with true density, SoF density index, and Strength superimposed (Threshold 90, Weighted).

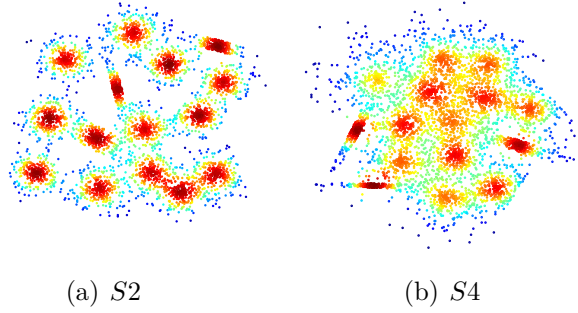


Figure 4.15: S-Sets datasets with SoF density index superimposed (Threshold 95, Weighted).

Figures concerning S-Sets and Newgroup graphs show only results obtained for weighted graphs, as those results are essentially identical for unweighted graphs.

The results concerning the Grid dataset are displayed on Figure 4.19, and show that the SoF density index is able to distinguish two areas with different densities, even when the density is constant on large areas of the graph. The distinction is even clearer when the parameter has higher values ($\theta > 10$), the SoF density score showing a more sharp jump from low to high density. This may be due to the smoothing happening when θ has a low value (higher-cost forests have higher probability to appear, and involve a larger

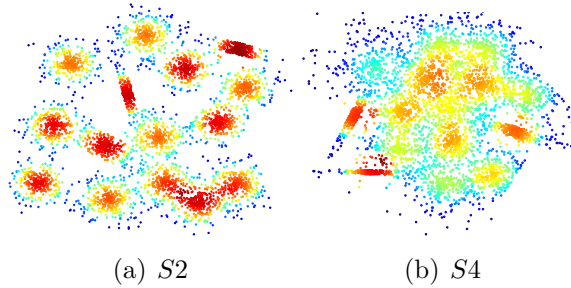


Figure 4.16: S-Sets datasets with strength superimposed (Threshold 95, Weighted).

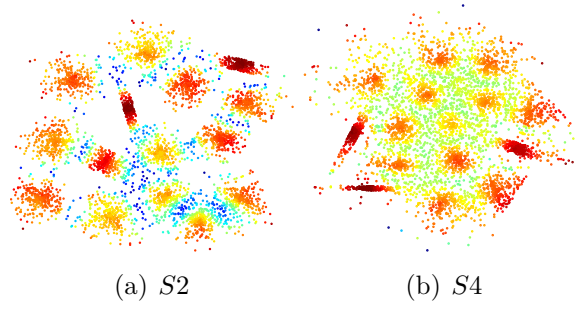


Figure 4.17: S-Sets datasets with CC superimposed (Threshold 95, Weighted).

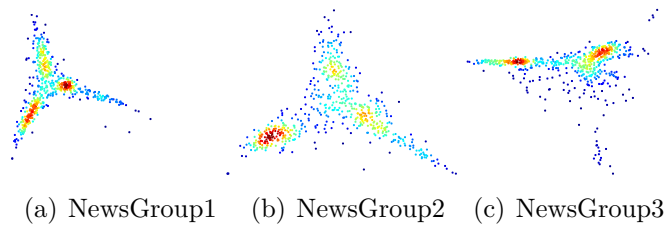


Figure 4.18: NewsGroup datasets with SoF density index superimposed (Threshold 95, Weighted).

part of the graph (see Section 4.5.4.1 for more details)). The results are mainly the same whatever the way a graph is constructed (Nearest Neighbors, Threshold method, Weighted or not), and only results for the graph constructed with 95-threshold and weighted with the affinities between nodes is shown.

4.5.4.3 k -Core results

In order to evaluate to which extent the high density areas are well identified with the generalized k -cores, we first draw curves representing the evolution of the true density's mean value on the graph, while removing the node having the lowest density measure. The closer a curve is to the true density curve, the more likely the corresponding measure identifies well the dense zones on the graph. In order to have a quantifiable measure of how close is a density measure to the true density, we also compute an “area under curve” (AUC) score, defined as the area below the curve of the corresponding measure, compared to the area below the true density curve. A score of one indicates that the measure identifies perfectly dense zones. The results obtained with the SoF density index are compared with another measure used as a baseline: the strength.

Figure 4.20 and 4.21 display mean density curves for different measures, for 10C-S1 and 10C-S2 graphs constructed with a 90 percentile threshold on top of the figure, and k -nearest neighbors with $k = 50$ on the bottom. The true density is represented by the

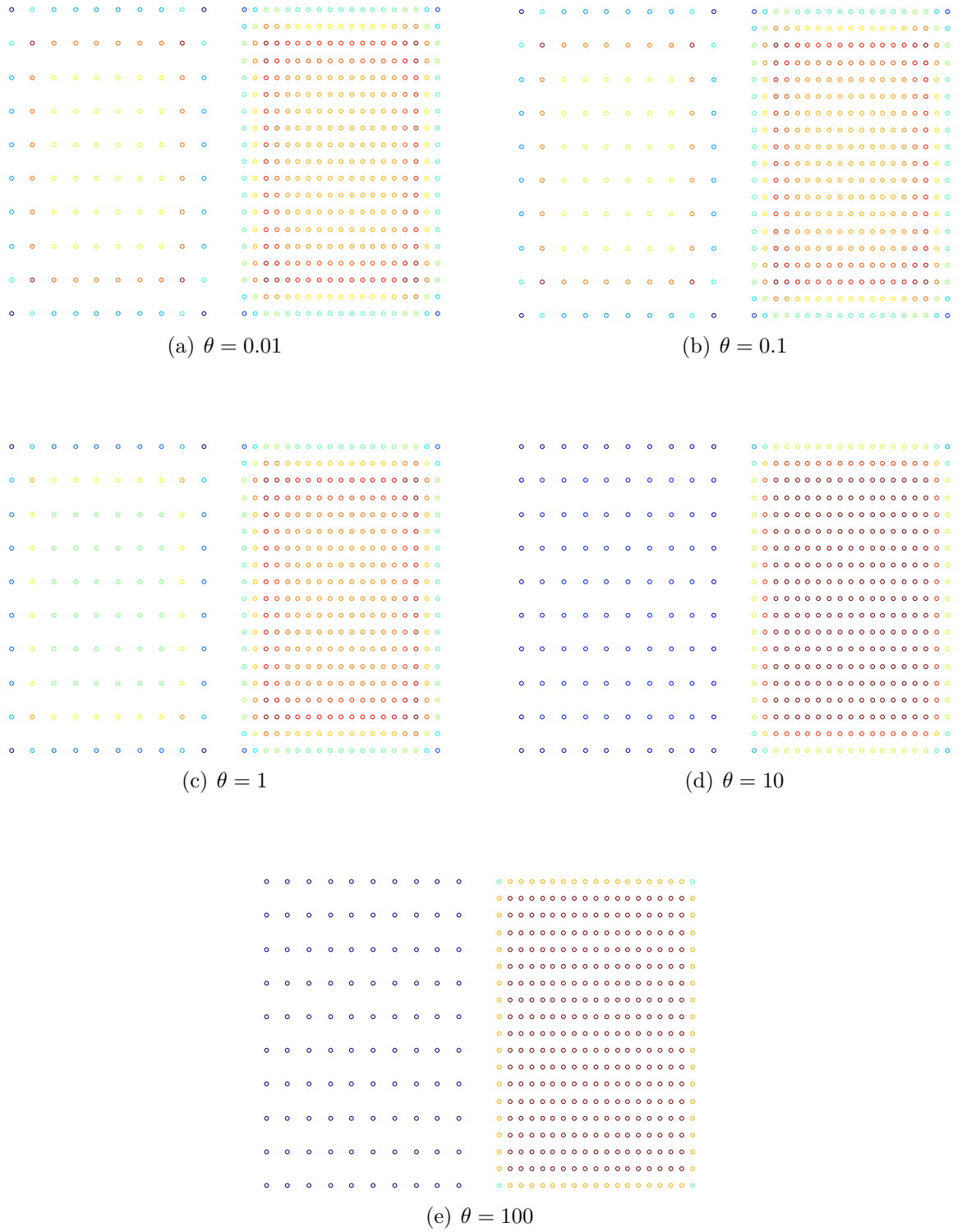


Figure 4.19: Grid dataset with SoF density index superimposed (Threshold 95, Weighted).

		thetaSoF	th90	th95	th99	NN5	NN50
10C-S1	SoF	0.01	0.7939	0.9613	0.8373	0.6501	0.7420
		0.1	0.8089	0.9618	0.8417	0.6499	0.7421
		1	0.9087	0.9670	0.8865	0.6490	0.7435
		10	0.9950	0.9955	0.9891	0.7068	0.8832
		100	0.9861	0.9861	0.9842	0.8117	0.9715
	Strength		0.7558	0.9934	0.9890	0.6973	0.7565
10C-S2	SoF	0.01	0.9046	0.9573	0.8276	0.6939	0.7782
		0.1	0.9100	0.9579	0.8311	0.6939	0.7780
		1	0.9502	0.9641	0.8715	0.6943	0.7787
		10	0.9924	0.9931	0.9874	0.7513	0.9366
		100	0.9702	0.9702	0.9699	0.8299	0.9655
	Strength		0.8610	0.9790	0.9873	0.7306	0.7743

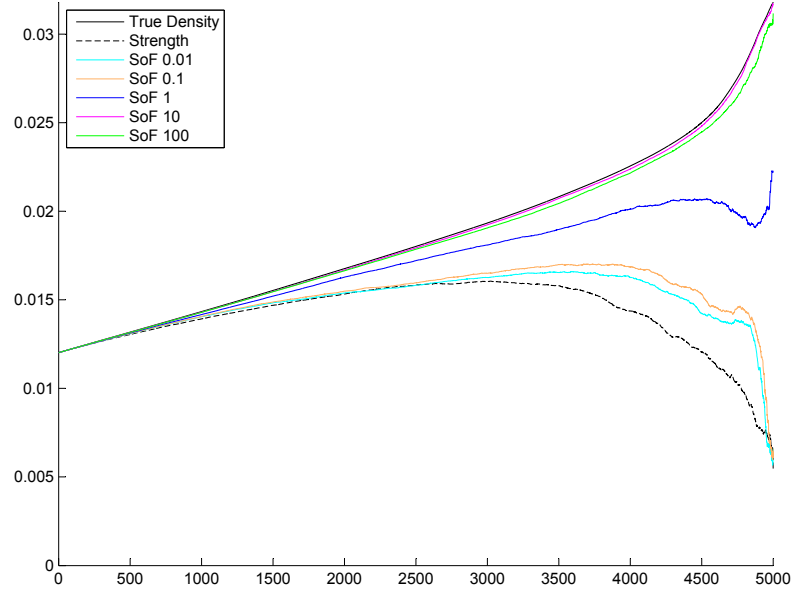
 Table 4.11: “Area” score for the generalized k -core procedure on 10C-S1 and 10C-S2.

black curve, and the strength by the dashed black line. Other colored curves stand for the SoF density index, each one for a particular value of the parameter θ . The different curves confirm the choice of the optimal value for the parameter θ , as the curves which are the closest to the true density curve are those related to $\theta = 10$ and $\theta = 100$. The 10-value leads to better results with 10C-S2 constructed with the threshold method, while the 100-value leads to better results with graphs constructed with nearest neighbor method. Moreover, let us note that these two parameter values give good results in the two cases.

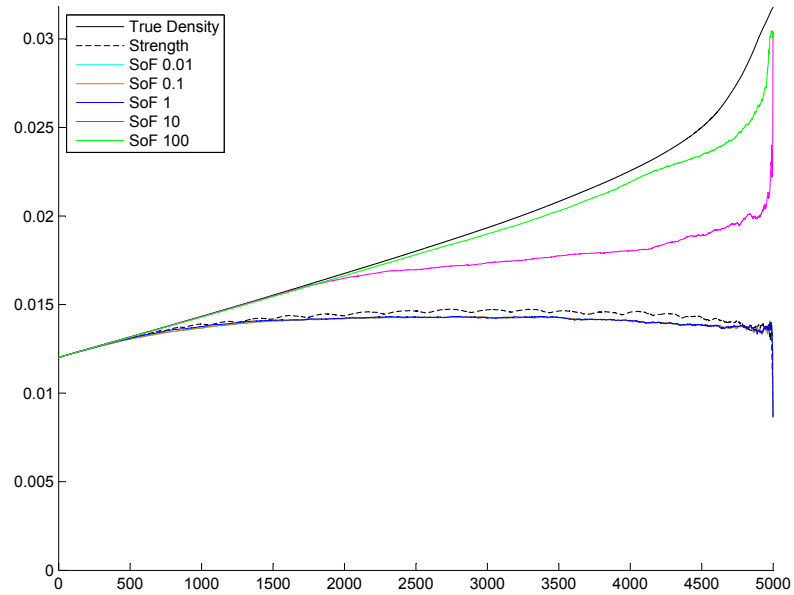
We can see that the strength fails to identify dense areas on these graphs, for the two graph construction methods, as the density curve completely diverges from the true curve. On the other hand, the SoF density index performs well, mainly for high values of θ . This is confirmed by the “area” score displayed in Table 4.11, where, for instance, strength obtain a score of 0.7558 for 10C-S1 for threshold graph, and the SoF with $\theta = 10$ a score of 0.9950 (the other “area” scores for the 10-communities datasets are displayed in Table 4.11).

Figure 4.22 illustrates how the generalized k-Core works, removing nodes presenting the lowest density measure. Three iterations are displayed, showing that the algorithm converges correctly to dense areas.

When looking at the “AUC” scores in Table 4.11, we can see that when using the k-NN for constructing graphs, the SoF density index is clearly superior to the strength, for high values of θ . This may be explained by the fact that the information concerning the connectivity is useless in this case, as all the nodes have theoretically almost the same degree. The SoF density index then makes a better use of the affinities borne by the edges of the graphs than the strength does.

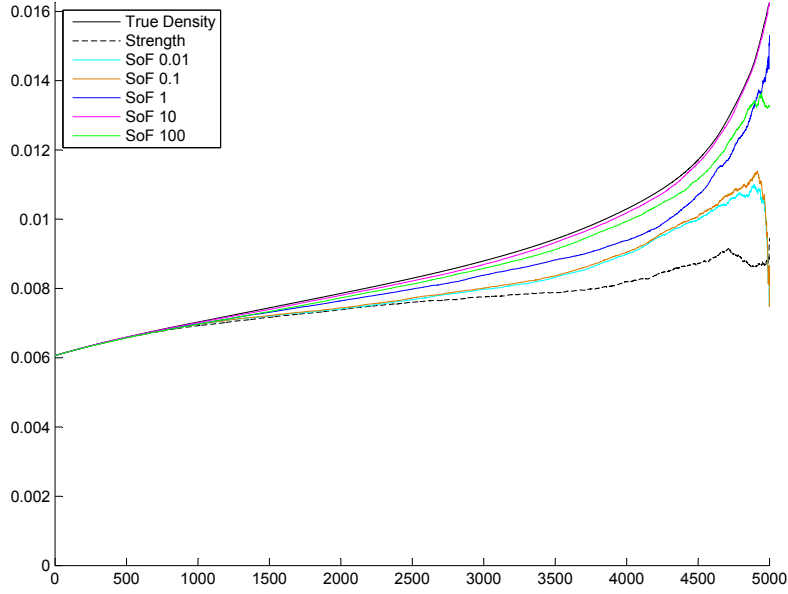


(a) 10C-S1 with th90

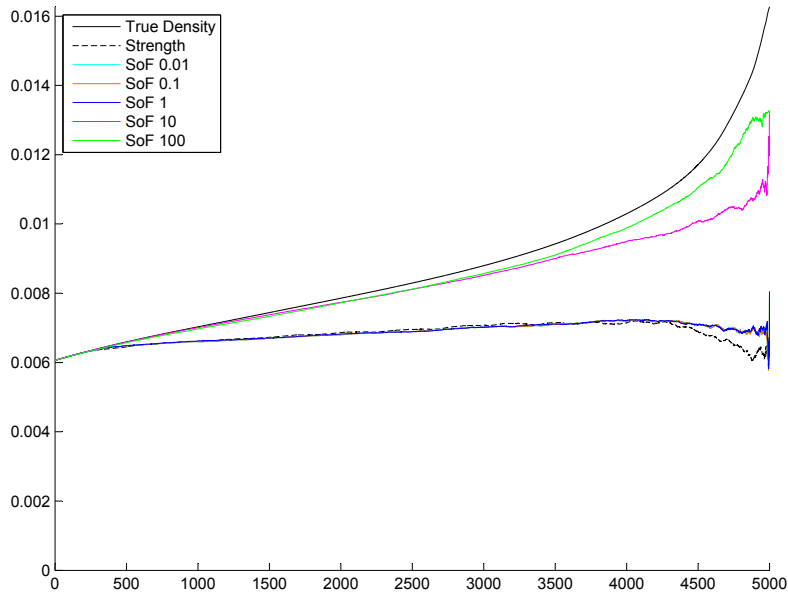


(b) 10C-S1 with NN50

Figure 4.20: Generalized k -Core curve for 10C-S1, constructed with 90-percentile threshold method (a) and k -Nearest Neighbors with $k = 50$ (b). X-axis represents the iteration number, while the Y-axis represents the value of the true mean density.



(a) 10C-S2 with th90



(b) 10C-S2 with NN50

Figure 4.21: Generalized k -Core curve for 10C-S2, constructed with 90-percentile threshold method (a) and k -Nearest Neighbors with $k = 50$ (b). X-axis represents the iteration number, while the Y-axis represents the value of the true mean density.

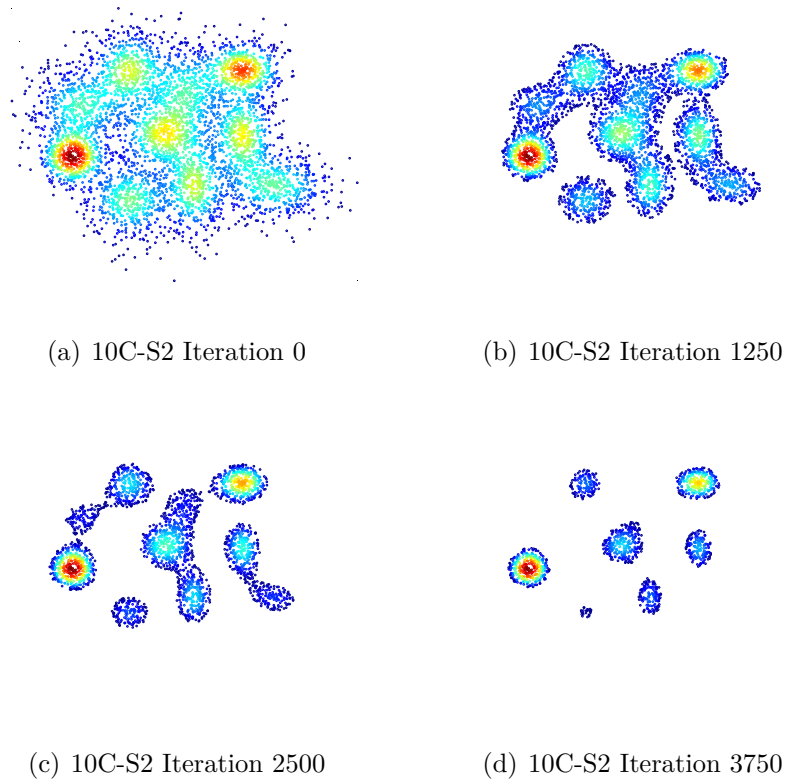


Figure 4.22: Different iterations of the k-Core procedure used to identify dense regions on 10C-S2.

10C S2	θ				
	0,01	0,1	1	10	100
Th90	2,0153	2,0051	1,8686	0,1900	0,4728
NN50	1,2473	1,3443	2,1291	1,4783	0,3428

Table 4.12: Skewness absolute values for the SoF density index distribution on the nodes of the 10-communities datasets (high sigma values S2), for different θ values.

When using the ϵ -graph construction method, the results show that the SoF index is clearly better for low threshold (ϵ) values. Indeed, the number of neighbors increases dramatically when the threshold is low, and again the information of connectivity becomes useless, nodes having each a large degree value. The only useful information are the affinities and, as before, the SoF density index uses it in a more efficient way than the strength. When the threshold is higher the two measures converge to the same value.

Parameter tuning

In Section 5.3.1.2, we fixed the value of θ to the one giving a SoF density score for which the correlation with the “true” density is the highest. Following arguments detailed in [122, 152], we then tested the skewness of the SoF density index’s distribution on the nodes of several graphs (10-communities dataset with high sigma values, constructed with threshold and nearest neighbors methods). The Figures 4.25 and 4.26 illustrates the effect of the value of θ on the skewness. It appears that the values giving the results that are the most correlated with the “true” density are almost always the ones for which the skewness is the smallest (in absolute value), i.e., when the distribution is the most symmetric. This procedure could be used as a data-adaptive way of choosing θ .

Alternatively, the choice of θ can be left to the user, allowing him to identify the kind of desired communities (small or large). Indeed, as already stated above, the value of the parameter θ smoothes the SoF density index on the nodes of the graph (see Figures 4.23 and 4.24 for an illustration of the smoothing effect). When θ values are small, as the trees used in the computation of the density index are large, dense areas are more spread and tend to merge, forming bigger dense regions. On the contrary, high values of θ favor low-cost forests, and thus assess the density index of a node on more local areas of the graph around that node. The dense areas are then smaller and more separated. This effect can also be seen as a scaling, aiming at identifying dense regions (or communities) of different scales.

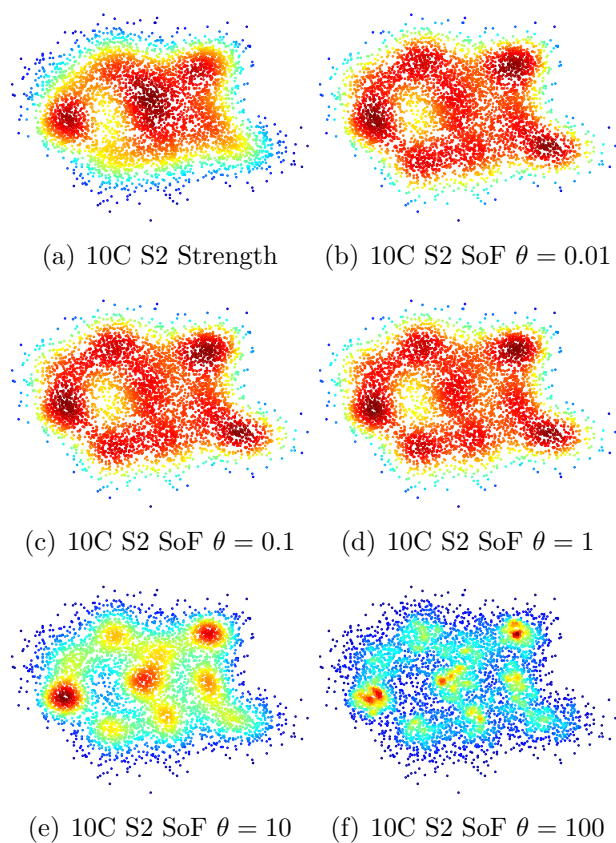


Figure 4.23: 10-communities datasets (high sigma values S2) with Strength and SoF density index superimposed (Threshold 90, Weighted).

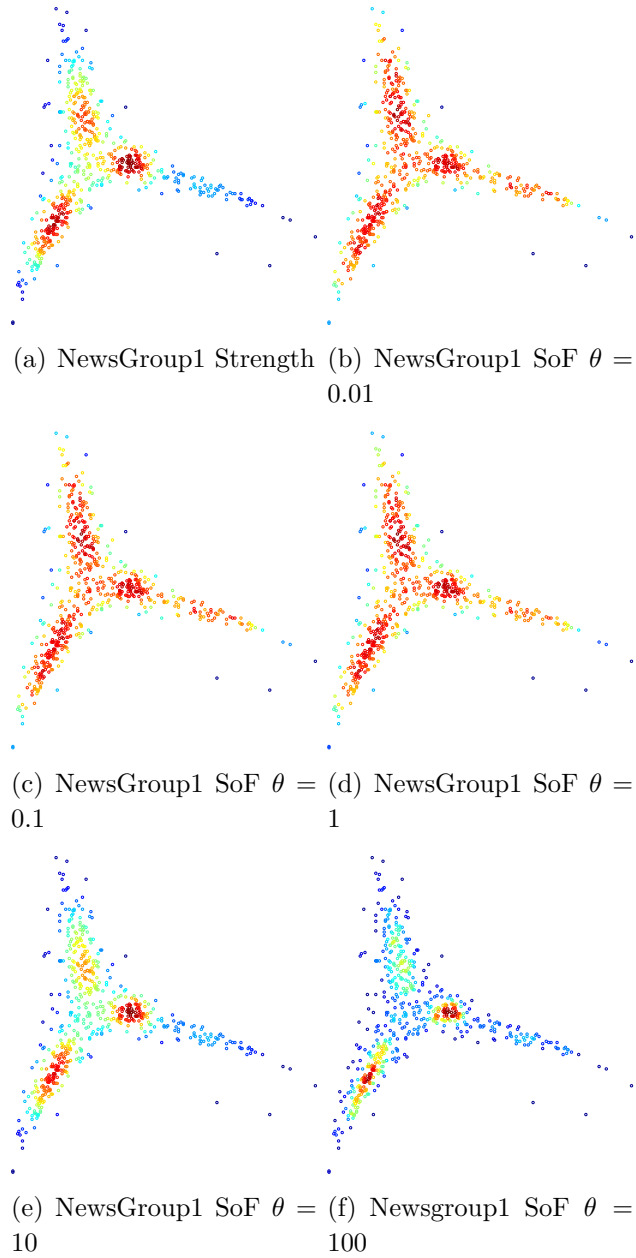


Figure 4.24: NewsGroup1 datasets with Strength and SoF density index superimposed (Threshold 90, Weighted).

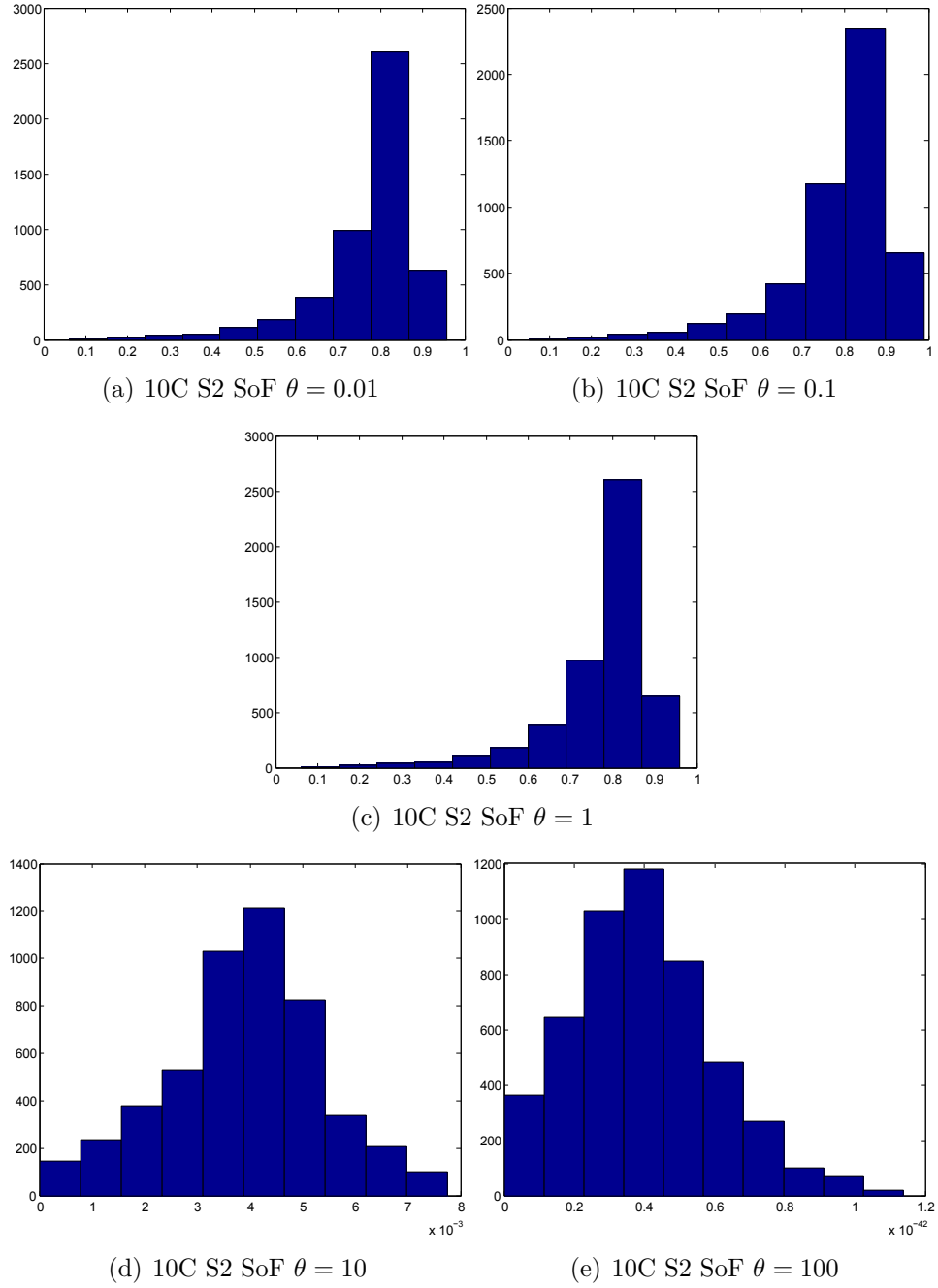


Figure 4.25: SoF density index distribution on the nodes of the 10-communities dataset (high sigma values S2) (Threshold 90, Weighted).

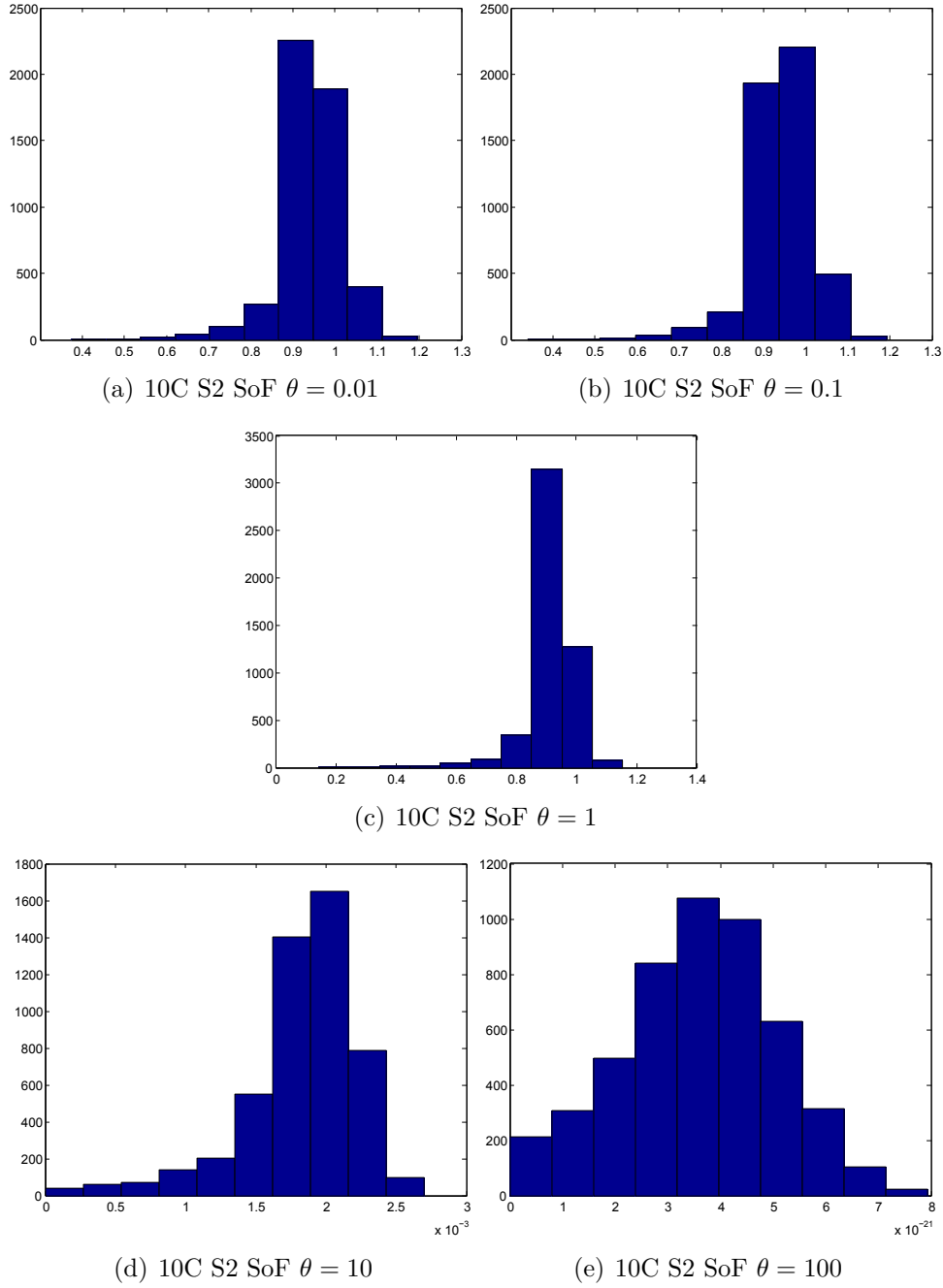


Figure 4.26: SoF density index distribution on the nodes of the 10-communities dataset (high sigma values S2) (Nearest Neighbors 50, Weighted).

4.6 Chapter conclusion

We introduced in this chapter a new density index on the nodes of a graph. The main idea behind the model is that a node has a high density index if it is present on a large number of (preferably low-cost) forests, together with a high outdegree. This model depends on a meta-parameter θ , biasing gradually the forests probabilities from uniform towards low-cost forests. A sum-over-paths statistical physics framework is used in order to derive the form of the index in terms of the immediate costs defined on the arcs. This density index is assessed in various ways: (1) using correlations between the SoF index and the “true” density, (2) visualizing the density scores on the nodes by superimposing it as a color on a graph plot, and (3) using a k -Core procedure, removing nodes with the lowest density score in order to converge to dense regions on the graph. The parameter θ can be tuned by the user, choosing the desired size of the researched dense areas, but we also propose a data-adaptive way for choosing a reasonable value of the parameter.

The application of the SoF density index to the task of searching dense areas on graphs shows that it performs well, being able to recover all the high density regions – corresponding to the center of clusters – on different graphs. Moreover, the correlation results between the SoF density index and the true density (when available) are often close to one. The SoF density index also gives more stable results than the strength regarding the way a graph is constructed.

Finally, the SoF density index can be computed efficiently by inverting a $n \times n$ matrix, where n is the number of nodes, leading to an overall time complexity of $\mathcal{O}(n^3)$. However, in the future, we probably could modify the proposed technique for working on large graphs, as in [106]. Indeed, the Sum-over-Forests measure only depends on the diagonal of the inverse matrix $\mathbf{Z}$ (this can be easily deduced from Equation (4.25). Moreover, the matrix $(\mathbf{I} + \mathbf{L}(\mathbf{W}))$ is diagonally dominant). In this case, scalable methods can be used for computing the diagonal of $\mathbf{Z}$ (see, e.g., [39, 41, 153]).

Chapter 5

The Sum-over-Forests clustering on graphs

Contents

5.1	Review of density-based clustering	109
5.2	The SoF divisive clustering	112
5.2.1	Experiments	113
5.2.2	Results	113
5.3	The SoF mode-seeking clustering	114
5.3.1	Experiments	119
5.3.2	Results	121
5.4	Chapter conclusion	122

This chapter introduces novel ways to identify dense regions in a graph based on (i) a divisive algorithm or (ii) a mode-seeking clustering technique, both relying on the Sum-Over-Forests density index as a density estimator.

5.1 Review of density-based clustering

This section provides a short survey of the related work concerning density-based clustering.

Clustering on graphs, also called community detection, is a topic that received a lot of attention recently, and extensive reviews exist on the subject (see for instance [47, 173]). We will review here some of the most used clustering techniques based on density.

DBSCAN [42] (Density-Based Spatial Clustering of Applications with Noise) is one of the most common clustering algorithms. The key idea of that method is that for each point of a cluster, its neighborhood of a given radius has to contain at least a minimum number of points, i.e., the density in the neighborhood has to exceed some threshold. Its definition of a cluster is based on the notion of density reachability, i.e., a point j is directly density-reachable from a point i if it is not farther away than a given distance, and if i is surrounded by sufficiently many points such that one may consider that i and j are part of a cluster. j is called density-reachable from i if there is a sequence of $i_1, \dots, i_n$ points with $i_1 = i$ and $i_n = j$ where each i_{k+1} is directly density-reachable from i_k . Note that the relation of density-reachability is not symmetric, so the notion of density-connectedness is introduced: two points i and j are density-connected if there is a point o such that o and i as well as o and j are density-reachable. So, a cluster must satisfy two properties: (1) all points within the cluster are pairwise density-connected and (2) if a point is density-connected to any point of the cluster, it is part of the cluster as well. DBSCAN does not require the number of clusters in the data to be specified, and can find arbitrarily shaped clusters.

OPTICS [6] (Ordering PoinTs to Identify the Clustering Structure) algorithm is based on the DBSCAN method, and improves two major weaknesses of the original method: extracting meaningful clusters, and cluster data sets well with large differences in densities. To do so, the points of the database are reordered in order that spatially closest points become neighbors in the ordering.

DENGRAPH is also inspired by the algorithm DBSCAN. Falkowski et al. [43] propose an adaptation of DBSCAN to the density based graph clustering, with their algorithm DENGRAPH. The intention behind it is to cluster similar nodes in a graph into communities. The density-based approach applies a local cluster criterion. Clusters are defined as regions in the graph in which the nodes are dense, and which are separated by regions of low node density. To detect regions of higher density, DENGRAPH computes neighborhoods which have a certain radius and must contain a minimum of nodes to be sure that the neighborhood is dense. A node having such a neighborhood is termed a core node. Nodes not having this kind of neighborhood are either border nodes if they are in the neighborhood of a core node or noise nodes. In order to build a cluster, DENGRAPH explores the graph by randomly picking nodes and puts all density connected nodes it encounters into the same cluster. If a node is not density connected to the nodes seen thus far, it is assigned to the next cluster candidate.

DENCLUE [65] (DENsity-based CLUstEring) is a clustering algorithm using a set of density distribution functions. It is built on the following ideas: (1) the influence of each data point is modeled using a mathematical function, called an influence function, which describes the impact of a data point within its neighborhood; (2) the global density of the data space can be modeled as the sum of the influence function applied to all data points;

(3) clusters are thus determined mathematically by identifying density attractors, which are local maxima of the global density function. Data points are assigned to clusters by hill climbing, i.e., points going to the same local maximum are put into the same cluster. A second version of this method, DENCLUE 2.0 [64], overcome the disadvantage of DENCLUE, that is the used hill climbing may make unnecessary small steps in the beginning and may never converge exactly to the maximum.

Mode-seeking methods are similar to the DENCLUE method, identifying modes (peaks of density) on datasets, like the **Mean Shift** algorithm [80], which compute the modes of a probability density function to find high density areas, using for instance Kernel density estimation (or Parzen Window). Mean shift associates each point with the nearby peak of the probability density function on the dataset. Then, for each data point, Mean shift defines a region around it and computes the mean of the data points. Then it shifts the center of the region to the mean and repeats the algorithm until it converges. After each iteration, we can consider that the region shifts to a more denser area of the dataset.

These methods were originally intended to be used in the feature space of the data, but adaptations to graph data were recently proposed [26, 43, 75, 94]. [75] introduced the median graph shift for clustering a set of graphs, based on the concept of the median graph, which can be defined as the graph that has the minimum sum of distances to all graphs in the set. It can be seen as the representative of the set [73]. [94] presented the graph shift algorithm for dense subgraph detection which alternates graph expansion and contraction. In these two methods, a mode is determined as a graph among a given set of graphs [75] or as a subgraph among the power set of a given graph [94], and their mode-seeking is done by shifting from one (sub)graph to another (sub)graph.

In this chapter, one of the clustering method we developed is quite similar to [26]. Indeed they use random walks on a graph to define modes (modes on a graph are the most frequently visited nodes among their local neighbors by random walks on the graph), coupled with a steepest ascent procedure to form the clusters.

A few other non-density-based clustering method are also reviewed hereunder, as they are used as state-of-the-art comparison in the experiments section.

Kernel hierarchical clustering [173] is an agglomerative clustering method gradually merging clusters. In a first step, a symmetric kernel on a graph is computed, such as the sigmoid commute-time kernel. This provides a similarity matrix between the nodes of the graphs, on which a hierarchical clustering is computed. The hierarchical clustering algorithm starts with one object per cluster (the withinclass inertia is zero) and then merges the clusters that lead to the smallest increase of the total within-cluster inertia. The process is repeated until all the groups are merged together and belong to the same cluster.

The **K-Means** [100] clustering algorithm is a very popular method used in pattern recognition, machine learning and data mining due to its simplicity and effectiveness. Basically, it computes the center for each cluster and assigns repeatedly the samples to the nearest cluster represented by the center. However, this method presents several drawbacks: the obligation to give the number k of clusters as an input, and the algorithm is in general not able to find non-convex clusters which makes the K-Means algorithm inadequate for complex non-linear data. The **Kernel K-Means** [138] problem is an extension of the k-means algorithm where the original data points are now mapped into a higher-dimensional feature space via a kernel function (as for example the sigmoid-commute time kernel). This “kernel trick” permits to separate non-linearly separable data in the original feature space.

The **Louvain Method** [14] calculates a partition of a given graph that maximizes the modularity (the modularity of any partition (clustering) measures the difference between the total number of edges connecting vertices that share a cluster and what would be expected in an equivalent graph with edges placed at random) by using the change in modularity when discovering a new node and adding it to an existing community. If the difference is not positive the node stays in the community it was assigned initially.

The following sections detail the two novel density-based algorithms, derived from the SoF density index.

5.2 The SoF divisive clustering

The first clustering method we developed uses a divisive approach, i.e., the data is initially considered as a single cluster and divided into several clusters as iterations go by. It is based on a simple k -Core procedure: the SoF density index is first computed on the nodes of the graphs used, and the nodes presenting the lowest density score are removed, along with the adjoining edges. The number of connected components (which are subgraphs where any two vertices are connected to each other by paths) is then computed. Starting from a single cluster, the number of components grows quite fast, depending on the number of nodes which are removed at each turn. The stopping criterion on this loop is to quit the iterative procedure when the number of connected components diminishes, revealing that we are starting to lose potential clusters. The connected components identified with this procedure are then considered as “cores”, i.e., center of clusters. We then determine the center of the “cores” as the node having the highest SoF density score. The clusters are then build by assigning each node of the graph to the nearby “core center”, using the $\mathbf{K}_{BoF}$ similarity measure.

5.2.1 Experiments

In this experimental section, the Sum-over-Forests divisive clustering algorithm is assessed on several graphs.

5.2.1.1 Datasets

The performance of the clustering algorithm is assessed on the 10-communities datasets. As a reminder, these datasets are artificial datasets we built: each one is made of ten clusters, created using Gaussian distributions $N(\mu, \sigma)$, μ being the mean (the center of the cluster) and σ^2 the variance of the data. Each cluster is made of 500 nodes, lying in two dimensions. The σ values are different in the two space directions, (x, y) . These values, called σ_x and σ_y are reported in Table 4.1 for the two sets: S_1 with small overlapping (i.e., with smaller σ values) and S_2 with strong overlapping (i.e., with higher σ values).

5.2.1.2 Evaluation methods

As a first evaluation, we test the ability of the algorithm to recover all the clusters that are present in the 10-communities datasets.

5.2.2 Results

The results of the divisive algorithm procedure applied on the 10-communities dataset with small overlapping (10C-S1) are displayed in Figure 5.1 where a few steps of the algorithm are shown. The subfigure displaying Step0 5.1(a) shows the original graphs, Step6 5.1(g) illustrates the peak of the number of connected components found, while we can see in the Step7 5.1(h) that the number of components is decreasing. The Step6 is then the step where the cores of the clusters are identified, and from which the clusters are formed. The final result of the clustering is displayed in Figure 5.2(a), where we can see that all the 10 original clusters are recovered. The result of the same procedure applied on the 10-communities dataset with large overlapping (10C-S2) is shown in Subfigure 5.2(b). However, when applying this procedure to the S4 dataset and removing the nodes iteratively, a problem appeared: the densities of the dense areas computed with the SoF density index are very different. Indeed, we can see on Figure 5.3(a) that 3 dense regions are much more dense than the others (the color superimposed on the nodes is dark red). Then when removing the nodes, the other dense areas disappear more quickly than those three. Those three peaks are shown on Figure 5.3(c). We thus tried to put all dense zones to approximately the same level of density, dividing the SoF index by its Laplacian-

smoothed version (see Section 4.1.3.1). Doing so allows to reduce the contrast between the dense areas, dividing the density score of a node with the one averaged with the score of its neighboring nodes. This acts as normalization of the density scores. The results of this procedure are shown on Figure 5.3(b) and 5.3(d), where the dense areas are now more “equally dense”. We then applied this procedure before using the divisive clustering algorithm on S-Set S4 dataset, and this let us recover all 15 clusters, as shown in Figure 5.4. This was not the case without the “normalization” step.

This algorithm led to promising results, and has an interesting property: it is indeed able to discover the core of the clusters, and thus not only performs a strict partitioning of the data. It is then able to perform partial clustering, for instance only selecting regions of the graph where the density is sufficiently high. However, as our goal was to develop a clustering algorithm being able to work directly on “natural graphs” (such as social networks), we tested our divisive algorithm on those kind of graphs, but the results were a bit deceiving. Indeed, most of the time, even when deleting many nodes, a lot of edges remained on the graph, leading to few connected components. We were then not able to recover the initial clusters, as with the artificial 10-communities datasets.

In order to overcome this problem, we then developed another algorithm, which is also based on the SoF density index, but using a mode-seeking procedure to form the clusters. This algorithm is detailed in the next section.

5.3 The SoF mode-seeking clustering

Our algorithm is based on a mode-seeking procedure [80], as used in the Mean-Shift method, with the difference that it operates directly on graphs instead of the feature space. This kind of method usually estimates the density on a dataset using Gaussian mixtures or kernel density estimation, and then follows a gradient ascent procedure to localize modes, which are local density maxima.

The first step of our SoF clustering algorithm is then to identify the modes of density on a graph. The density estimation is done by computing the Sum-over-Forests density index on each node. Then, starting from any node, a steepest ascent procedure is used, jumping from the original node to the one of its neighbor having the highest density score. This procedure ends when a local density peak (mode) is reached, i.e., when arriving on a node having the highest local SoF density score (i.e., if the current node, which is a local maximum, has no neighbor having a density score strictly superior, thus avoiding any cycle to appear if two nodes have the same density score). The nodes to where all the other nodes converge are identified as the modes. Those different modes are then considered as the prototypes or centers of the different clusters on the graph.

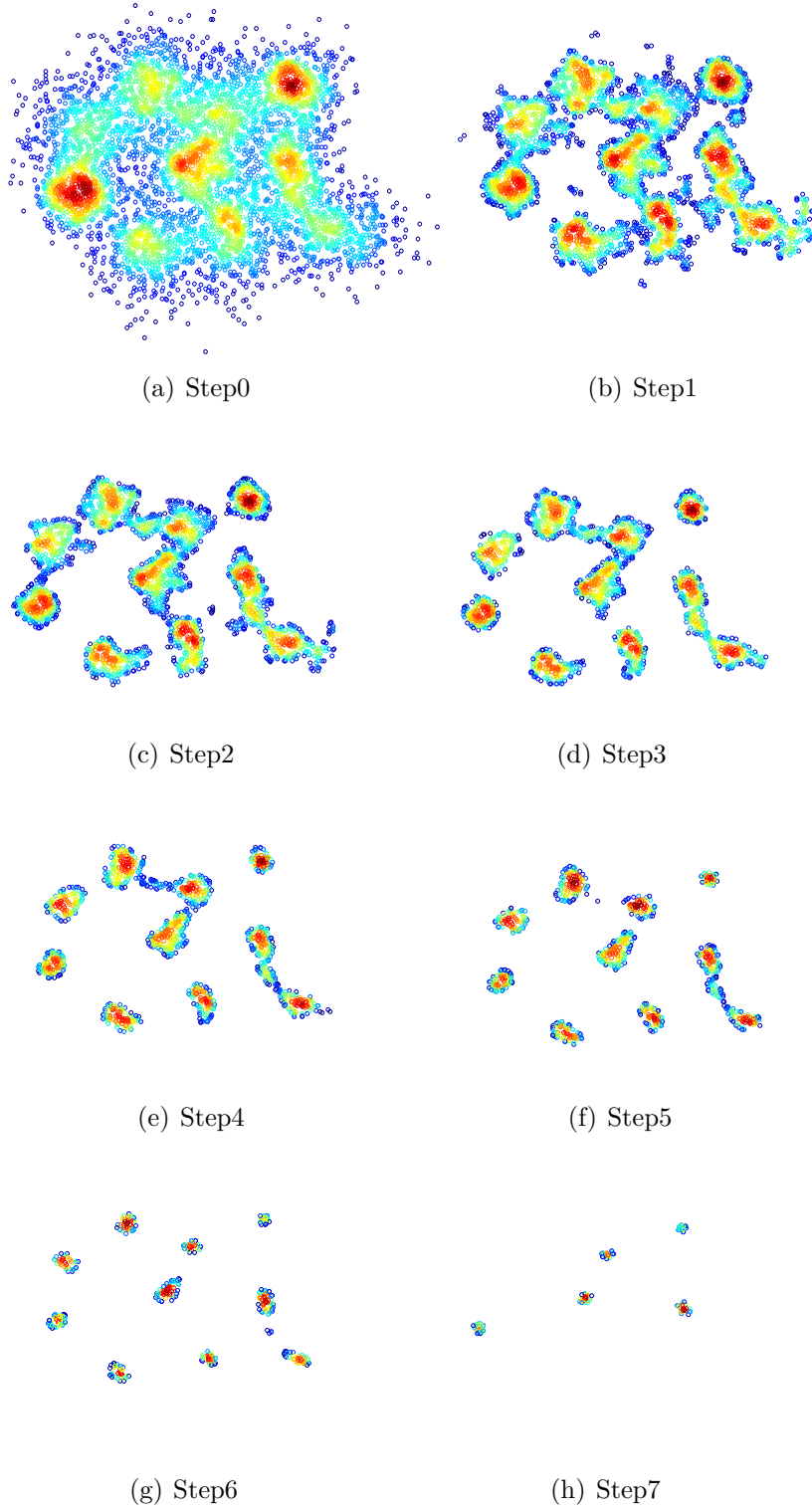
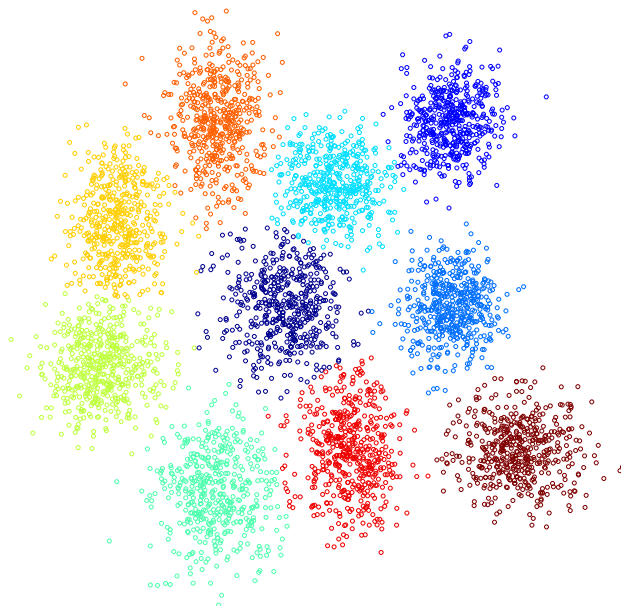
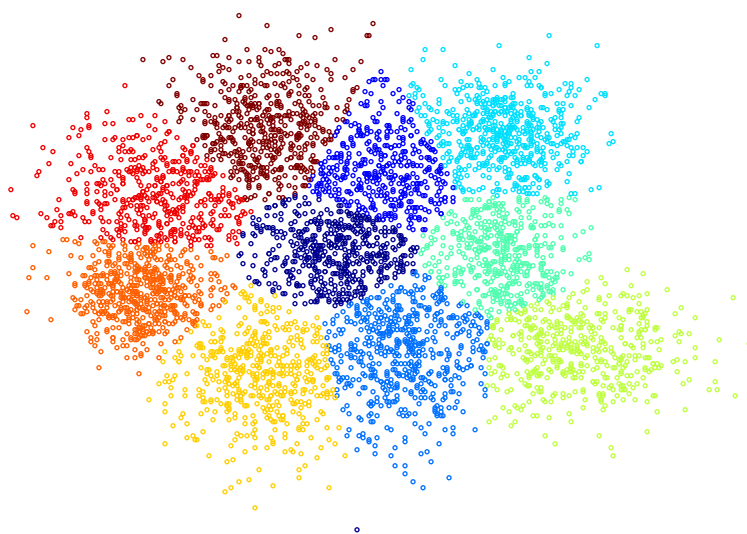


Figure 5.1: SoF divisive clustering on 10C-S1, constructed with 95-percentile threshold method, and SoF's parameter $\theta = 10$.



(a) 10C-S1



(b) 10C-S2

Figure 5.2: SoF clustering on 10C-S1 and 10C-S2, constructed with 95-percentile threshold method, and SoF's parameter $\theta = 10$. Each cluster is represented by a color.

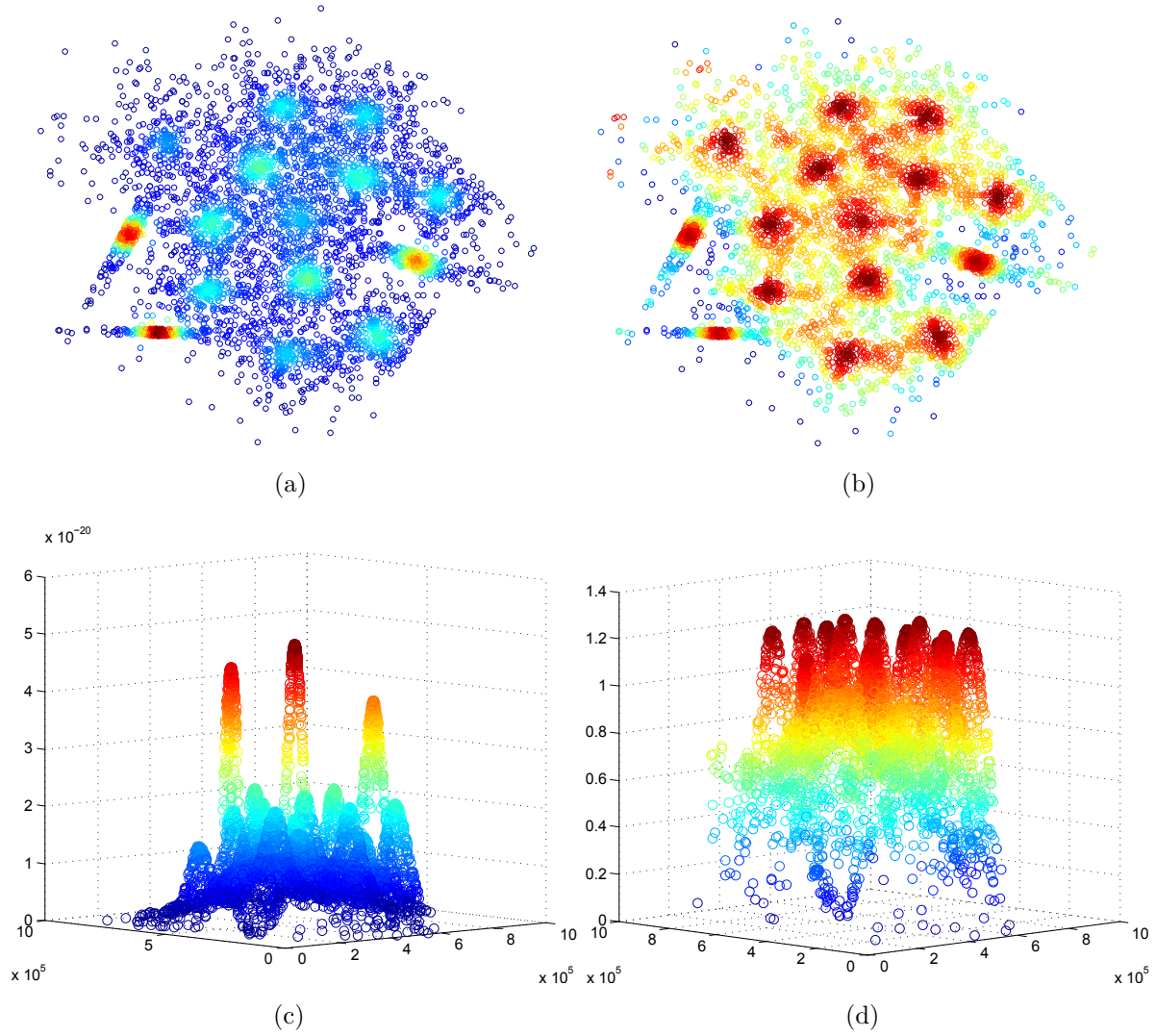


Figure 5.3: S-Set S4 dataset with SoF density index superimposed. (a) represents the original SoF density index, (b) the density index divided by its smoothed version, (c) the original density index in 3D, and (d) the modified density index in 3D. Z-axis for (c) and (d) represents the magnitude of the density indices.

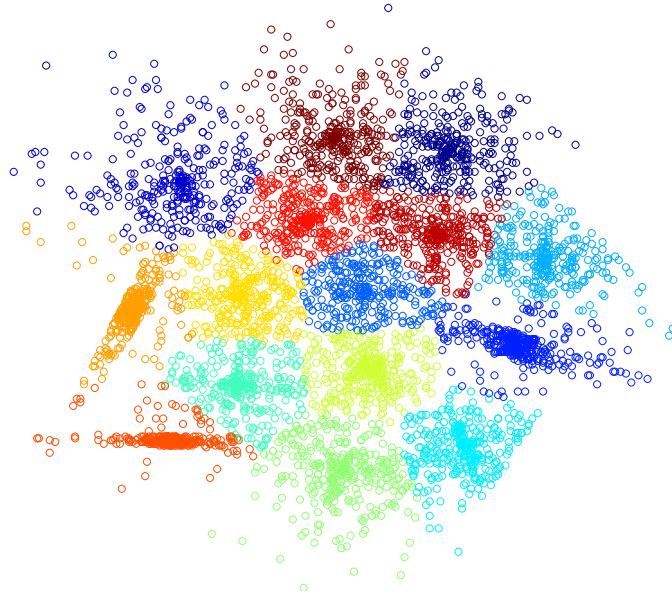


Figure 5.4: SoF clustering on S-Set S4 dataset, constructed with 95-percentile threshold method, and SoF’s parameter $\theta = 10$. Each cluster is represented by a color.

In order to form the different clusters, each node is associated to its corresponding mode, which can be done in different ways. The simplest one is maybe to associate a node to the mode it converges to, as computed during the mode seeking part of the algorithm. Another way to proceed, which gives better results, is to compute a similarity measure between each node and the different modes, and to associate a node to the most similar mode. We tested several similarity measure, as the cosine similarity, the Katz matrix [79], etc. The one showing the best results (detailed in section 5.3.2) is the K_{BoF} similarity measure described in Chapter 3.

This clustering method works naturally when the number of clusters is not known a priori, but can also work when this number is known. Indeed, when the set of modes has been identified, it can be restrained to the m clusters truly present in the data, by selecting the m largest modes to where most of the nodes converge, and then applying the clustering procedure. Obviously, this can only be done if the number of modes found is greater than m . Moreover, in that case, the association of a node to a mode cannot be done by assigning that node to the node (mode) it converges to, as some modes are removed.

Concerning the complexity of the clustering algorithms, the main bottleneck is the computation of the SoF density index, which has a complexity of $\mathcal{O}(n^3)$. Indeed, when using the divisive clustering algorithm, only few steps are performed in order to remove the lowest density nodes, computing the connected components at each turn, which has $\mathcal{O}(n + e)$ complexity, n being the number of nodes and e the number of edges [154]. The mode-seeking algorithm examines all n nodes as starting points, and reaches density

maxima which may take theoretically $(n - 1)$ steps, leading to an $n(n - 1)$ complexity (but in practice there are only a few steps from the starting point to the local maxima). Those two algorithms then have an overall $\mathcal{O}(n^3)$ complexity.

In summary, this clustering method has two main benefits: it does not need to know the number of clusters a-priori (even if it is also able to work when it is known), and works directly on graphs.

5.3.1 Experiments

In this experimental section, the Sum-over-Forests mode seeking clustering is performed on several “real” (i.e., not artificial) graphs.

5.3.1.1 Datasets

Five small to moderately sized networks data are used to assess the performance of the clustering algorithm:

- Zachary’s Karate Club: A social network of friendships between members of a karate club at a US university in the 1970s [176],
- Dolphins: An undirected social network of frequent associations between dolphins in a community living in New Zealand [98],
- Football: Network of American football games during regular season Fall 2000 [116],
- Sampson’s Monastery: This dataset comes from an ethnographic study of community structure in a New England monastery [134],
- Political Books: A network of books about politics published for the 2004 US presidential election [81].

Table 5.1 displays the main features of those graphs. All networks have binary edges. Monastery dataset used a +1/-1 edge weight in order to reflect the like/dislike relationship between monks, but -1 affinities have been set to zero, in order not to have negative affinities.

5.3.1.2 Evaluation methods

In the clustering task, the ground-truth clustering for each of the network used is known. This let us compare the clustering obtained by the SoF method to the true clustering, using

	# nodes	# edges	# communities
Zachary	34	156	2
Dolphins	62	318	2
Football	115	1226	12
PolBooks	105	882	3
Monastery	18	33	4

Table 5.1: Characteristics of the five networks used in clustering task.

two different criteria: the adjusted rand index, and the normalized mutual information. We compare our clustering method without the information about the number of clusters to the Louvain method [14], and we also test the case where we give in input the number of clusters, comparing with the Kernel K-Means and the Kernel Hierarchical clustering from Yen [173]. Kernel K-Means is derived from the standard K-Means clustering algorithm and is able to identify nonlinearly separable clusters, with the help of a kernel matrix. Kernel Hierarchical clustering is a kernel version of Ward’s hierarchical clustering [167]. As these two latter methods need a kernel matrix as input, we use the sigmoid commute-time kernel, with a parameter $\gamma = 7$, which has proven to give good results [173]. Note that, in order to perform fair comparisons between the methods, we first perform, before starting the clustering, a random mixing of the data, which may explain lower results than in literature for some algorithms, especially for the Louvain method. In order to measure the quality of the clustering obtained with our SoF method, we compared it to the “true” clustering, as this was known for every datasets used. The Adjusted Rand Index (ARI) was first used, which is an extension of the Rand Index [125]. This index is a measure of the similarity between two data clusterings and can be defined as the following: Given a set of n elements $S = \{o_1, \dots, o_n\}$ and two partitions of S to compare: $X = \{X_1, \dots, X_r\}$, a partition of S into r subsets, and $Y = \{Y_1, \dots, Y_s\}$, a partition of S into s subsets, define the following:

- a, the number of pairs of elements in S that are in the same set in X and in the same set in Y
- b, the number of pairs of elements in S that are in different sets in X and in different sets in Y
- c, the number of pairs of elements in S that are in the same set in X and in different sets in Y
- d, the number of pairs of elements in S that are in different sets in X and in the same set in Y

The Rand index, R , is then

$$R = \frac{a + b}{a + b + c + d}. \quad (5.1)$$

The Rand index has however a problem when two random clusterings are compared: they do not have a constant index, for example zero. [68] therefore introduced the adjusted Rand index, based on the assumption that the process is the generalized hypergeometric distribution, i.e., the ideal and computed clustering are selected at random so that the number of objects in both clustering is fixed.

Another index was also used, the Normalized Mutual Information (NMI) criterion [151]. Mutual information, which is a symmetric measure to quantify the statistical information shared between two distributions, provides a reliable indication of the shared information between a pair of clusterings:

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}}, \quad (5.2)$$

where X and Y are random variables described by two cluster labellings, $I(X, Y)$ denote the mutual information between X and Y , and $H(X)$ denote the entropy of X .

5.3.2 Results

The results for the adjusted rand index (ARI) and the normalized mutual information criterion (NMI) are displayed in Table 5.2. The parameter θ for the SoF clustering method is fixed to 0.1, as this value provides in all cases the best results. The results for Kernel K-Means, Kernel Hierarchical clustering and Louvain method are averaged on 100 runs. The SoF clustering is used together with the $\mathbf{K}_{\text{BoF}}$ similarity measure to form the different clusters.

On the Zachary dataset, we can see that the SoF clustering obtains a perfect score of one in the two criteria, even when the number of classes is not provided. Kernel K-means and Hierarchical clustering obtain good results, with knowledge of the number of classes. The Louvain method finds three clusters (instead of two) and its result is quite below the other methods. For the Dolphins dataset, results without providing the number of clusters are essentially the same for the SoF and Louvain method. When the number of clusters is known, the SoF method's results jump to the ones of Kernel K-Means and above the ones of Hierarchical clustering. In the Football case, where 12 clusters are to be found, the SoF method identifies 11 clusters, and the Louvain method 6, which results in higher score for the SoF. When the number of clusters is provided, the results do not change, because the number of clusters found by the SoF is below the true number, and we cannot

		SoF+nbClust	Kernel K-Means	Hierarchical	SoF	Louvain
Zachary (2)	ARI	1	0.9252	0.8823	1 (2)	0.5943 (3)
	NMI	1	0.9067	0.8365	1 (2)	0.6360 (3)
Dolphins (2)	ARI	0.9348	0.9294	0.7537	0.4080 (4)	0.4293 (3)
	NMI	0.8889	0.8833	0.7014	0.6168 (4)	0.5985 (3)
Football (12)	ARI	0.5874	0.7709	0.8893	0.5874 (11)	0.3572 (6)
	NMI	0.7505	0.8669	0.9269	0.7505 (11)	0.6337 (6)
PolBooks (3)	ARI	0.6679	0.6703	0.6559	0.6679 (2)	0.5974 (4)
	NMI	0.6102	0.5706	0.5522	0.6102 (2)	0.5218 (4)
Monastery (4)	ARI	0.5352	0.4223	0.1088	0.2475 (8)	0.4686 (5)
	NMI	0.6548	0.5756	0.3579	0.6247 (8)	0.6332 (5)

Table 5.2: Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) values for five networks. SoF+nbClust represents the SoF clustering method with number of clusters in input. When the number of clusters is not known (SoF and Louvain), the number of clusters found are given in parenthesis.

add artificially new clusters (in the Dolphins case, we only keep the modes which attract the most nodes). Hierarchical clustering gives the best results on this dataset. For the Political Books dataset, the SoF clustering obtains the best results in all cases, clearly above the Louvain method, and slightly above the K-Means and Hierarchical clustering. Finally, the results for the Monastery dataset show an ARI score inferior to the Louvain method, but a NMI score almost equal for these two methods. When the number of clusters is provided, the SoF clustering performs better than all other methods.

We also performed clustering tests on the 10-communities datasets. In the 10C-S1 case, the ten clusters originally present in the dataset are recovered, and this is confirmed by an NMI (resp. ARand) score of 0.9408 (resp. 0.9422). In the 10C-S2 case, eight datasets on ten are recovered, but as the assignment of the nodes to the corresponding mode do not allow overlapping of clusters, the NMI (resp. ARand) score is here only of 0.6946 (resp. 0.5822), despite the fact that the algorithm identifies well most center of the clusters.

5.4 Chapter conclusion

Two new clustering algorithms have been proposed: the SoF divisive clustering, and the SoF mode-seeking clustering.

The divisive algorithm is based on a k-Core procedure, deleting nodes of the graph used presenting the lowest density scores. Connected components are then identified,

modelling the “cores” of the clusters. Clusters are finally formed by assigning each node of the graph to the nearby center of a cluster. While leading to promising results, and being able to recover “cores” of clusters, the inability of this algorithm to work on “true” graphs (like social networks) directed us to develop another type of clustering algorithm, also based on density.

The SoF mode-seeking clustering algorithm is based on a mode-seeking procedure, identifying the modes as peaks of SoF density, and clustering the datasets using a forest similarity between the nodes. This method performs well on the five networks shown here. When the number of clusters is not known, this algorithm matches, and in some cases, outperforms the Louvain Method, and even the Kernel K-Means and the Hierarchical clustering. When the number of clusters is supplied, the results can greatly increase.

In the future, the mode-seeking procedure detailed in this work, using the SoF density index, could be coupled to another clustering algorithm. Indeed, the modes identified could serve as cluster prototypes given as input to, for instance, the K-Means method.

General conclusion

Main research results

In this thesis, we were mainly interested in two problems: (1) comparing graphs, and nodes of graphs, and (2) discovering dense regions on graphs.

The graph comparison problem already received a lot of attention these last years, notably among the class of graph comparison methods called graph kernels. Those kernels often compute similarities between subgraphs (walks, paths, trees, ...) to determine an overall similarity between graphs. Based on classical kernels, we developed three new kernels: (1) the Modified Weisfeiler-Lehman kernel, which is modification of the original Weisfeiler-Lehman kernel [147] but using a Jensen-Shannon divergence to compare distribution of labels, (2) the randomized shortest-path kernel, based on the shortest-path kernel [15] but here using a variation of the shortest-path distance introducing entropy and authorizing more exploration on the graph, and (3) the graph-features kernel, which computes linear-time features on graphs, building feature vectors which are compared using a linear kernel. Although these three kernels provide interesting results, none of them overcame the Weisfeiler-Lehman kernel in term of accuracy. However, the graph features kernel was designed for its speed and thus surpasses all the other kernels in test, and provides good results in terms of accuracy on three datasets on four.

The following chapter of this thesis introduced some new similarity measures between nodes of a graph. Those new measures are based on the matrix-forest theorem from Chebotarev [21]. The first measure developed, which is called the Bag-of-Forests (BoF) similarity measure, assigns a Boltzmann probability distribution on the forests of the graph, favoring low-cost ("small") forests to be sampled at random. It is based on a simple and intuitive idea: if two nodes are simultaneously present on low-cost forests, they are likely to be similar. Two distances were derived from this BoF similarity measure: (1) the BoF potential distance and (2) the BoF logarithmic distance, which are then turned back again into similarities. Those measures are then tested on a semi-supervised classification task on several datasets (9 Newsgroup and 4 WebKb Cocite) against state-of-art methods (including Regularized Commute-Time Kernel $\mathbf{K}_{RCT}$, Regularized Laplacian Kernel $\mathbf{K}_{RL}$, Bag-of-Paths (BoP) betweenness,...). The original BoF similarity measures do not perform as well as the top methods (especially $\mathbf{K}_{RCT}$ and BoP), but the SoF potential and the

SoF logarithmic similarity measures give more than average results on the NewsGroup datasets, and the best results on the WebKB datasets.

From the BoF similarity measure, we then derived a measure of density on nodes of a graph: the Sum-over-Forests (SoF) density index. Our goal is to identify dense regions on graphs, suggesting some community-like structures. Repeating the same procedure than in the similarity measure, a Boltzmann probability is again assigned to forests. The idea is also quite similar as above: a node receives a high density score if it is present on many (preferably low-cost) forests. This density index, which depends on a parameter θ , is assessed in several ways: (1) a visual way, allowing to check on a plot of the graph if the dense areas are well identified, superimposing the density as a color on the graph plot; (2) using correlations between the density scores of the nodes and the “true” density, which is known for graphs we built; (3) a k -core procedure, removing the nodes with the lowest density scores, and seeing if the procedure converges to the densest areas on the graph. All these assessment methods show that the SoF density index is a promising density measure, being able to recover all dense areas on several graphs. Using the θ parameter, the user is also able to choose the desired size of the communities he would like to find, going from large, merged communities to smaller ones. We also provided a data-adaptive way of choosing θ , based on the skewness of the distribution of the SoF density scores.

Finally, using the SoF density index, two density-based clustering methods were developed. The first one is a divisive SoF clustering algorithm, working in a k -core fashion, removing the nodes presenting the lowest density scores, and revealing iteratively connected components, which are considered as the center of the dense areas on the graph. Those connected components also serve as center of the researched clusters. The divisive algorithm has however the drawback that if a graph is strongly connected, no new connected components appears. The second clustering method is based on a mode-seeking procedure [80], defining modes of a graph as peaks of density, which are here modelled as peaks of SoF density scores. For those two clustering methods, the clusters are finally formed by assigning each node to its nearest cluster center, using for instance the BoF similarity measures. The clusterings delivered by the mode seeking clustering algorithm are then assessed on several graphs for which a “ground-truth” clustering is known, and the correspondence between the two is then compared using the adjusted rand index and the normalized mutual information criterion. The main advantages of these developed algorithms are that they works directly on graphs and they do not need a priori the number of clusters (but these can be provided to improve the results). Given the results obtained, we can state that the mode-seeking algorithm performs well on most of the datasets used, compared to state-of-the-art methods like the Louvain Method, the Kernel K-means and the Hierarchical clustering. When the number of clusters is not specified a priori, it equals the Louvain Method, and sometimes the Kernel K-means and the Hierarchical clustering. When the number of clusters is provided, the results can be greatly increased.

Applications

As already stated in Chapter 2, graph kernels have important applications in bioinformatics, and in particular in the classification of molecules. Indeed, the graph kernels (which can be seen as a similarity measure between graphs) developed in this work could be used to classify molecules with regards to their carcinogenic character or not, to select potential good molecules among the millions created by combinatorial chemistry (see Section 2.1.2), etc. The capacity to “screen” molecules via computers increases greatly the efficiency of new drugs discovery, which may help companies to realize enormous savings.

The similarity measures between nodes of a graph, developed in Chapter 3, have been applied to semi-supervised classification in this work. The aim of this type of classification is to infer the unknown labels on certain nodes, knowing the true labels on other nodes. The similarity measure between two nodes is then crucial to estimate if a node will have a certain label, based for instance on the labels of its neighbors. It could be applied in social network in order to predict unknown attributes about people in the network (such as age, sex, political orientation, etc.). Other applications of nodes similarity are the recommendation systems, where a graph is constituted of certain nodes representing people and other nodes representing items. Based on a similarity between people nodes (or item nodes), it is possible to recommend for instance an item that a person like to another person which is close (where the closeness is quantified by node similarities) to him. This is a very frequently used method in online merchant websites, such as Amazon [110].

The discovery of dense regions on graphs is of primary interest in social networks, telecommunication networks, etc. Indeed, the dense areas may suggest some community-like structure, like group of friends, community of people with same political views, etc. The SoF density index can therefore be used to reveal regions on graphs which are more dense (here, nodes which belong to the same tree in a forest) from the other, or even displaying the cores of dense areas, which are the densest parts of a graph. Moreover, this method does not require an exact partitioning of the graph, and may even extract only regions where the density is above a certain threshold, for instance only groups of people who are strongly linked. In a marketing point of view, identifying dense subgraphs on, for instance, telecommunication networks could allow to send to the members of the community target advertisements, or to better design and target offers on special rates for calling to a limited number of prespecified phone numbers.

Contrarily to identifying dense regions on graphs, clustering on graphs tries to produce an exact partition of the data, whether the number of clusters is known in advance or not. It can be used on social networks, or in customer segmentation, to group people in clusters according to their age, sex, . . . , again to produce specific targetted advertisements.

When trying to index documents (as text, webpage, . . .), clustering algorithms can also be used to group those documents according to the field they belong to (as the NewsGroup dataset for instance). The tools developed here identify the densest areas on a graph with the SoF density index, and use them as center of clusters, to which all other nodes are associated according to their distance to the closest cluster center.

Future prospects

In the future, some research perspectives exist for the different new notions developed in this work. The first could be an investigation of the SoF density index on large graphs, as in [106]. Indeed, this density index has currently a time complexity of $\mathcal{O}(n^3)$, and an improvement of the computation time could lead to an application to graphs with large number of nodes. A second perspective would be to modify the SoF index on directed graphs in order to take into account converging forests instead of (or in addition to) diverging ones. Two types of dense regions could then be revealed : sources (with high outdegree) and targets (with high indegree). Thirdly, the SoF mode-seeking algorithm could be used as a preprocessing step for other clustering algorithms which need to define the number of clusters a priori (like the K-Means, for instance). The SoF mode-seeking algorithm would then propose modes on the studied graph and the associated algorithm would form the clusters.

Bibliography

- [1] ABELLO, J., RESENDE, M. G., SUDARSKY, S., AND SUDARSKY, R. Massive quasi-clique detection. In *Proceedings of the 5th Latin American Theoretical Informatics Symposium* (2002), LATIN '02, pp. 598–612. [\[73\]](#), [\[74\]](#)
- [2] ABNEY, S. *Semisupervised Learning for Computational Linguistics*, 1st ed. Chapman & Hall/CRC, 2007. [\[52\]](#)
- [3] AGGARWAL, C., AND WANG, H. *Managing and Mining Graph Data*. Advances in database systems. Springer, 2010. [\[2\]](#), [\[3\]](#), [\[71\]](#)
- [4] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases* (1994), VLDB '94, pp. 487–499. [\[17\]](#)
- [5] AKAMATSU, T. Cyclic flows, markov process and stochastic traffic assignment. *Transportation Research B* 30, 5 (1996), 369–386. [\[72\]](#)
- [6] ANKERST, M., BREUNIG, M. M., KRIEGEL, H.-P., AND SANDER, J. Optics: ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD international conference on Management of data* (1999), SIGMOD '99, pp. 49–60. [\[110\]](#)
- [7] BABAI, L., AND CODENOTTI, P. Isomorphism of hypergraphs of low rank in moderately exponential time. In *Proceeding of the 49th Annual IEEE Symposium on Foundation of Computer Science* (2008), FOCS '08, IEEE Computer Society, pp. 667–676. [\[1\]](#)
- [8] BAJORATH, J. Integration of virtual and high-throughput screening. *Nature Reviews Drug Discovery* 1, 11 (2002), 882–894. [\[14\]](#)
- [9] BALAS, E., AND YU, C. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing* 15, 4 (1986), 1054–1068. [\[73\]](#)
- [10] BALASUNDARAM, B., BUTENKO, S., AND HICKS, I. V. Clique relaxations in social network analysis: The maximum k-plex problem. *Operation Research* 59, 1 (Jan. 2011), 133–142. [\[74\]](#)

- [11] BARRAT, A., BARTHELEMY, M., PASTOR-SATORRAS, R., AND VESPIGNANI, A. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America* 101, 11 (2004), 3747–3752. [\[76\]](#)
- [12] BATAGELJ, V., AND ZAVERSENIK, M. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification* 5, 2 (2011), 129–145. [\[2\]](#), [\[74\]](#), [\[75\]](#)
- [13] BLONDEL, V., GAJARDO, A., HEYMANS, M., SENELLART, P., AND VAN DOOREN, P. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *Siam Review* (2004), 647–666. [\[43\]](#)
- [14] BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R., AND LEFEBVRE, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 10 (2008), P10008. [\[3\]](#), [\[112\]](#), [\[120\]](#)
- [15] BORGWARDT, K. M., AND KRIEGEL, H.-P. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining* (2005), ICDM '05, IEEE Computer Society, pp. 74–81. [\[1\]](#), [\[22\]](#), [\[25\]](#), [\[28\]](#), [\[29\]](#), [\[35\]](#), [\[125\]](#)
- [16] BRANDES, U., AND ERLEBACH, T. *Network analysis: methodological foundations*. Springer-Verlag, 2005. [\[73\]](#), [\[74\]](#)
- [17] BRIGHTWELL, G., AND WINKLER, P. Maximum hitting time for random walks on graphs. *Random Structures and Algorithms* 1, 3 (Oct. 1990), 263–276. [\[42\]](#)
- [18] BRON, C., AND KERBOSCH, J. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM* 16, 9 (Sept. 1973), 575–577. [\[73\]](#)
- [19] CHANG, C.-C., AND LIN, C.-J. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 3 (2011), 27. [\[2\]](#)
- [20] CHAPPELLE, O., SCHÖLKOPF, B., AND ZIEN, A., Eds. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. [\[52\]](#)
- [21] CHEBOTAREV, P. A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics* 159, 5 (2011), 295–302. [\[46\]](#), [\[51\]](#), [\[55\]](#), [\[79\]](#), [\[125\]](#)
- [22] CHEBOTAREV, P. The walk distances in graphs. *Discrete Applied Mathematics* 160, 10–11 (2012), 1484 – 1500. [\[79\]](#)
- [23] CHEBOTAREV, P., AND AGAEV, R. Forest matrices around the laplacian matrix. *Linear Algebra and its Applications* 356 (2002), 253–274. [\[49\]](#), [\[50\]](#), [\[72\]](#), [\[78\]](#)

-
- [24] CHEBOTAREV, P., AND SHAMIS, E. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control* 58, 9 (1997), 1505–1514. [\[1\]](#), [\[3\]](#), [\[44\]](#), [\[47\]](#), [\[49\]](#), [\[68\]](#), [\[72\]](#), [\[77\]](#), [\[78\]](#)
- [25] CHEBOTAREV, P., AND SHAMIS, E. On proximity measures for graph vertices. *Automation and Remote Control* 59, 10 (1998), 1443–1459. [\[44\]](#), [\[47\]](#), [\[68\]](#)
- [26] CHO, M., AND LEE, K. M. Mode-seeking on graphs via random walks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition* (2012), CVPR ’12, IEEE Computer Society, pp. 606–613. [\[111\]](#)
- [27] CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. *Introduction to algorithms, 2nd Edition*. The MIT Press, 2001. [\[80\]](#)
- [28] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297. [\[18\]](#), [\[20\]](#)
- [29] COX, T., AND COX, M. *Multidimensional scaling, 2nd ed.* Chapman and Hall, 2001. [\[51\]](#)
- [30] CRISTIANINI, N., AND SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. [\[18\]](#)
- [31] DE BORDA, J. C. Mémoire sur les élections au scrutin. [\[55\]](#)
- [32] DEBNATH, A. K., LOPEZ DE COMPADRE, R. L., DEBNATH, G., SHUSTERMAN, A. J., AND HANSCH, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34, 2 (1991), 786–797. [\[32\]](#)
- [33] DEZA, M. M., AND LAURENT, M. *Geometry of Cuts and Metrics*, 1st ed. Springer Publishing Company, Incorporated, 2009. [\[51\]](#)
- [34] DICE, L. R. Measures of the amount of ecologic association between species. *Ecology* 26, 3 (July 1945), 297–302. [\[41\]](#)
- [35] DIESTEL, R. *Graph Theory, 4th Edition*, vol. 173 of *Graduate texts in mathematics*. Springer, 2012. [\[5\]](#)
- [36] DIMASI, J. A., HANSEN, R. W., AND GRABOWSKI, H. G. The price of innovation: new estimates of drug development costs. *Journal of Health Economics* 22, 2 (Mar. 2003), 151–185. [\[13\]](#)
- [37] DINESHKUMAR, B., KUMAR, P. V., BHUVANESHWARAN, S., AND MITRA, A. Advanced drug designing softwares and their applications in medical research. *International Journal of Pharmacy & Pharmaceutical Sciences* 2, 3 (2010). [\[13\]](#)

- [38] DOBSON, P. D., AND DOIG, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783. [\[32\]](#)
- [39] DUFF, I., ERISMAN, A., AND REID, J. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986. [\[108\]](#)
- [40] ECKERT, H., AND BAJORATH, J. Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches. *Drug Discovery Today* 12, 5–6 (2007), 225–233. [\[14\]](#)
- [41] ERISMAN, A. M., AND TINNEY, W. F. On computing certain elements of the inverse of a sparse matrix. *Communications of the ACM* 18, 3 (1975), 177–179. [\[108\]](#)
- [42] ESTER, M., KRIEGEL, H., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (1996), KDD '96, AAAI Press, pp. 226–31. [\[110\]](#)
- [43] FALKOWSKI, T., BARTH, A., AND SPILIOPOULOU, M. Dengraph: A density-based community detection algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence* (2007), WIC '07, pp. 112–115. [\[110\]](#), [\[111\]](#)
- [44] FELZENSZWALB, P. F., AND HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision* 59, 2 (2004), 167–181. [\[3\]](#)
- [45] FLAKE, G. W., LAWRENCE, S., AND GILES, C. L. Efficient identification of web communities. SIGKDD '00, pp. 150–160. [\[72\]](#)
- [46] FOGGIA, P., SANSONE, C., AND VENTO, M. A performance comparison of five algorithms for graph isomorphism. In *Proceedings of the IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition* (2001), GbR '01, pp. 188–199. [\[15\]](#)
- [47] FORTUNATO, S. Community detection in graphs. *Physics Reports* 486 (2010), 75–174. [\[109\]](#)
- [48] FOUSS, F., FRANÇOISSE, K., YEN, L., PIROTTE, A., AND SAERENS, M. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural Networks* 31 (2012), 53–72. [\[43\]](#), [\[44\]](#), [\[45\]](#)

-
- [49] FOUSS, F., PIROTTE, A., RENDERS, J.-M., AND SAERENS, M. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering* 19, 3 (2007), 355–369. [\[45\]](#)
- [50] FOUSS, F., SAERENS, M., AND SHIMBO, M. *Algorithms and Models for Network Data & Link Analysis*. In preparation. [\[38\]](#), [\[43\]](#), [\[75\]](#), [\[76\]](#)
- [51] FOUSS, F., YEN, L., PIROTTE, A., AND SAERENS, M. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proceedings of the 6th International Conference on Data Mining* (2006), ICDM '06, pp. 863–868. [\[3\]](#), [\[43\]](#), [\[46\]](#)
- [52] FRANÇOISSE, K., KIVIMAKI, I., MANTRACH, A., ROSSI, F., AND SAERENS, M. A bag-of-paths framework for network data analysis. *Submitted for publication; available on ArXiv as ArXiv:1302.6766* (2013), 1–36. [\[46\]](#), [\[47\]](#), [\[51\]](#)
- [53] FRANTI, P., AND VIRMAJOKI, O. Iterative shrinking method for clustering problems. *Pattern Recognition* 39(5) (2006), 761–765. [\[81\]](#)
- [54] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. [\[15\]](#)
- [55] GÄRTNER, T., FLACH, P., AND WROBEL, S. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, B. Schölkopf and M. Warmuth, Eds., vol. 2777 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 129–143. [\[22\]](#)
- [56] GAÜZÈRE, B., BRUN, L., VILLEMIN, D., AND BRUN, M. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In *Proceedings of the 21st International Conference on Pattern Recognitions* (2012), ICPR '12, pp. 1775–1778. [\[23\]](#)
- [57] GOETHALS, B. Frequent set mining. In *Data Mining and Knowledge Discovery Handbook*. Springer, 2010, pp. 321–338. [\[17\]](#)
- [58] GROSSO, A., LOCATELLI, M., AND CROCE, F. Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem. *Journal of Heuristics* 10, 2 (2004), 135–152. [\[73\]](#)
- [59] GUTMAN, I., AND TRINAJSTIĆ, N. Graph theory and molecular orbitals. total φ -electron energy of alternant hydrocarbons. *Chemical Physics Letters* 17, 4 (1972), 535–538. [\[16\]](#), [\[146\]](#)

- [60] HARCHAOUI, Z., AND BACH, F. Image classification with segmentation graph kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2007), CVPR '07, pp. 1–8. [\[22\]](#)
- [61] HARVILLE, D. A. *Matrix Algebra from a Statistician's Perspective*. Springer-Verlag, 1997. [\[79\]](#)
- [62] HAUSSLER, D. Convolution kernels on discrete structures. Tech. rep., Computer Science Department, UC Santa Cruz, 1999. [\[22\]](#)
- [63] HELMA, C., KING, R. D., KRAMER, S., AND SRINIVASAN, A. The predictive toxicology challenge 2000–2001. *Bioinformatics* 17, 1 (2001), 107–108. [\[32\]](#)
- [64] HINNEBURG, A., AND GABRIEL, H.-H. Denclue 2.0: Fast clustering based on kernel density estimation. In *Advances in Intelligent Data Analysis VII*, M. R. Berthold, J. Shawe-Taylor, and N. Lavra, Eds., vol. 4723 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 70–80. [\[111\]](#)
- [65] HINNEBURG, A., AND KEIM, D. A. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th ACM International Conference on Knowledge Discovery and Data Mining* (1998), KDD '98, pp. 58–65. [\[110\]](#)
- [66] HORVÁTH, T., GÄRTNER, T., AND WROBEL, S. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining* (2004), KDD '04, ACM, pp. 158–167. [\[23\]](#)
- [67] HUBERMAN, K., AND INFANTINO, R. High-throughput screening in industry. In *Anticancer Drug Development Guide*. Springer, 2004, pp. 23–39. [\[13\]](#), [\[14\]](#)
- [68] HUBERT, L., AND ARABIE, P. Comparing partitions. *Journal of Classification* 2, 1 (1985), 193–218. [\[121\]](#)
- [69] ITO, T., SHIMBO, M., KUDO, T., AND MATSUMOTO, Y. Application of kernels to link analysis. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining* (2005), KDD '05, pp. 586–592. [\[44\]](#)
- [70] JACCARD, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles* 37 (1901), 547–579. [\[40\]](#)
- [71] JAYNES, E. T. Information theory and statistical mechanics. *Physical Review* 106 (1957), 620–630. [\[78\]](#)
- [72] JEHL, G., AND WIDOM, J. Simrank: a measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining* (2002), KDD '02, pp. 538–543. [\[43\]](#)

-
- [73] JIANG, X., MÜNGER, A., AND BUNKE, H. On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 10 (2001), 1144–1151. [\[111\]](#)
 - [74] JOHNSON, R., AND WICHERN, D. *Applied Multivariate Statistical Analysis*, 5th Ed. Prentice Hall, 2002. [\[40\]](#)
 - [75] JOUILI, S., TABBONE, S., AND LACROIX, V. Median graph shift : A new clustering algorithm for graph domain. In *Proceedings of the 20th International Conference on Pattern Recognition* (2010), ICPR '10. [\[111\]](#)
 - [76] KANDOLA, J., CRISTIANINI, N., AND SHAW-ETAYLOR, J. Learning semantic similarity. *Advances in Neural Information Processing Systems (NIPS) 15* (2002), 657–664. [\[44\]](#)
 - [77] KASHIMA, H., TSUDA, K., AND INOKUCHI, A. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning* (2003), ICML '03, AAAI Press, pp. 321–328. [\[22\]](#), [\[23\]](#)
 - [78] KATAYAMA, K., HAMAMOTO, A., AND NARIHISA, H. An effective local search for the maximum clique problem. *Information Processing Letters* 95, 5 (2005), 503–511. [\[73\]](#)
 - [79] KATZ, L. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43. [\[42\]](#), [\[118\]](#)
 - [80] KOONTZ, W., NARENDRA, P., AND FUKUNAGA, K. A graph-theoretic approach to nonparametric cluster analysis. *IEEE Transactions on Computers* (1976), 936–944. [\[2\]](#), [\[111\]](#), [\[114\]](#), [\[126\]](#)
 - [81] KREBS, V. New political patterns. *Unpublished* (2004). [\[119\]](#)
 - [82] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86. [\[28\]](#)
 - [83] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., AND TOMKINS, A. The web and social networks. *Computer* 35, 11 (2002), 32–36. [\[72\]](#)
 - [84] LAFON, S., AND LEE, A. B. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 9 (2006), 1393–1403. [\[84\]](#)
 - [85] LANG, K. NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning* (1995), ICML '95, pp. 331–339. [\[52\]](#)

- [86] LEBICHOT, B., KIVIMÄKI, I., FRANÇOISSE, K., AND SAERENS, M. Semi-supervised classification through the bag-of-paths group betweenness. *CoRR abs/1210.4276* (2012). [\[52\]](#), [\[54\]](#)
- [87] LEE, V. E., RUAN, N., JIN, R., AND AGGARWAL, C. C. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. 2010, pp. 303–336. [\[73\]](#)
- [88] LEICHT, E. A., HOLME, P., AND NEWMAN, M. E. J. Vertex similarity in networks. *Phys. Rev. E* 73 (2006), 026120. [\[42\]](#)
- [89] LI, G., SEMERCI, M., YENER, B., AND ZAKI, M. J. Graph classification via topological and label attributes. In *Proceedings of the 9th Workshop on Mining and Learning with Graphs* (2011), MLG '11. [\[34\]](#)
- [90] LI, X., FOO, C., AND NG, S. Discovering protein complexes in dense reliable neighborhoods of protein interaction networks. *Computational Systems Bioinformatics Conference 6* (2007), 157–168. [\[72\]](#)
- [91] LIEBER, D. *Functional assays for screening antibody activity in droplet-based microfluidics*. PhD thesis, Strasbourg, 2010. [\[13\]](#)
- [92] LIN, J. Divergence measures based on the Shannon entropy. *Information Theory, IEEE Transactions on* 37, 1 (1991), 145–151. [\[28\]](#)
- [93] LIU, G., AND WONG, L. Effective Pruning Techniques for Mining Quasi-Cliques. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases* (2008), ECML PKDD '08, Springer-Verlag, pp. 33–49. [\[74\]](#)
- [94] LIU, H., AND YAN, S. Robust graph mode seeking by graph shift. In *Proceedings of the 27th International Conference on Machine Learning* (2010), ICML '10. [\[111\]](#)
- [95] LÜ, L., AND ZHOU, T. Link prediction in complex networks: A survey. *Physica A* 390, 6 (2011), 11501170. [\[41\]](#)
- [96] LUCE, D., AND PERRY, A. A method of matrix analysis of group structure. *Psychometrika* 14, 2 (1949), 95–116. [\[71\]](#), [\[73\]](#)
- [97] LUCE, R. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* 15, 2 (1950), 169–190. [\[74\]](#)
- [98] LUSSEAU, D., SCHNEIDER, K., BOISSEAU, O., HAASE, P., SLOOTEN, E., AND DAWSON, S. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology* (2003), 396–405. [\[119\]](#)

-
- [99] LUXBURG, U. V., AND ALAMGIR, M. Density estimation from unweighted k-nearest neighbor graphs: a roadmap. In *Advances in Neural Information Processing Systems 26*. 2013, pp. 225–233. [\[77\]](#)
- [100] MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), L. M. L. Cam and J. Neyman, Eds., vol. 1, University of California Press, pp. 281–297. [\[112\]](#)
- [101] MACSKASSY, S. A., AND PROVOST, F. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8 (2007), 935–983. [\[53\]](#)
- [102] MAHÉ, P. *Fonctions noyaux pour molécules et leur application au criblage virtuel par machines à vecteurs de support*. PhD thesis, École Nationale Supérieure des Mines de Paris, 2006. [\[13\]](#)
- [103] MAHÉ, P., UEDA, N., AKUTSU, T., PERRET, J.-L., AND VERT, J.-P. Extensions of marginalized graph kernels. In *Proceedings of the 21st international conference on Machine learning* (2004), ICML '04, ACM, pp. 70–. [\[22\]](#)
- [104] MAHÉ, P., AND VERT, J.-P. Graph kernels based on tree patterns for molecules. *Machine Learning* 75, 1 (2009), 3–35. [\[23\]](#)
- [105] MANLY, C. J., LOUISE-MAY, S., AND HAMMER, J. D. The impact of informatics and computational chemistry on synthesis and screening. *Drug discovery today* 6, 21 (2001), 1101–1110. [\[14\]](#)
- [106] MANTRACH, A., VAN ZEEBROECK, N., FRANCO, P., SHIMBO, M., BERSINI, H., AND SAERENS, M. Semi-supervised classification and betweenness computation on large, sparse, directed graphs. *Pattern Recognition* 44, 6 (2011), 1212 – 1224. [\[45\]](#), [\[108\]](#), [\[128\]](#)
- [107] MANTRACH, A., YEN, L., CALLUT, J., FRANCOISSE, K., SHIMBO, M., AND SAERENS, M. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), 1112–1126. [\[46\]](#), [\[72\]](#)
- [108] MARTINS, A. F. T., BICEGO, M., MURINO, V., AGUIAR, P. M. Q., AND FIGUEIREDO, M. A. T. Information theoretical Kernels for generative embeddings based on hidden Markov models. In *Proceedings of the 20th joint IAPR international conference on Structural, Syntactic, and Statistical Pattern Recognition* (2010), SSSPR '10, Springer-Verlag, pp. 463–472. [\[28\]](#)
- [109] MCCLOSKEY, B., AND HICKS, I. V. Combinatorial algorithms for the maximum k-plex problem. *Journal of combinatorial optimization* 23, 1 (2012), 29–49. [\[74\]](#)

- [110] MEYER, F. Recommender systems in industrial contexts. *CoRR abs/1203.4487* (2012). [\[3\]](#), [\[127\]](#)
- [111] MORGAN, H. L. The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *Journal of Chemical Documentation* 5, 2 (1965), 107–113. [\[16\]](#), [\[146\]](#)
- [112] NADLER, B., LAFON, S., COIFMAN, R., AND KEVREKIDIS, I. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advances in Neural Information Processing Systems (NIPS)* 18 (2005), 955–962. [\[84\]](#)
- [113] NEUHAUS, M., AND BUNKE, H. Automatic learning of cost functions for graph edit distance. *Inf. Sci.* 177, 1 (2007), 239–247. [\[15\]](#)
- [114] NEUHAUS, M., AND BUNKE, H. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007. [\[15\]](#)
- [115] NEWMAN, M. Clustering and preferential attachment in growing networks. *Physical Review E* 64, 2 (2001), 025102. [\[40\]](#)
- [116] NEWMAN, M., AND GIRVAN, M. Community structure in social and biological networks. *Proceedings of the National Academy Science* (2002), 7821–7826. [\[119\]](#)
- [117] NEWMAN, M. E. J. *Networks: An introduction*. Oxford University Press, 2010. [\[74\]](#)
- [118] NORRIS, J. R. *Markov Chains*. Cambridge University Press, 1997. [\[42\]](#)
- [119] PAN, J.-Y., YANG, H.-J., FALOUTSOS, C., AND DUYGULU, P. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the 10th ACM international conference on Knowledge Discovery and Data Mining* (2004), KDD '04, pp. 653–658. [\[45\]](#)
- [120] PATTILLO, J., VEREMYEV, A., BUTENKO, S., AND BOGINSKI, V. On the maximum quasi-clique problem. *Discrete Applied Mathematics* 161, 1–2 (2013), 244–257. [\[74\]](#)
- [121] PRŽULJ, N., WIGLE, D. A., AND JURISICA, I. Functional topology in a network of protein interactions. *Bioinformatics* 20, 3 (2004), 340–348. [\[72\]](#)
- [122] RADOVANOVIĆ, M., NANOPOULOS, A., AND IVANOVIĆ, M. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research* 11 (2010), 2487–2531. [\[103\]](#)
- [123] RALAIVOLA, L., SWAMIDASS, S. J., SAIGO, H., AND BALDI, P. Graph kernels for chemical informatics. *Neural Networks* 18, 8 (2005), 1093 – 1110. [\[22\]](#)

-
- [124] RAMON, J., AND GÄRTNER, T. Expressivity versus efficiency of graph kernels. In *Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences* (2003), MGTS '03, pp. 65–74. [\[23\]](#)
 - [125] RAND, W. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* 66, 336 (1971), 846–850. [\[120\]](#)
 - [126] RANDIC, M. Characterization of molecular branching. *Journal of the American Chemical Society* 97, 23 (1975), 6609–6615. [\[17\]](#), [\[146\]](#)
 - [127] RAVASZ, E., SOMERA, A. L., MONGRU, D. A., OLTVAI, Z. N., AND BARABÁSI, A. L. Hierarchical organization of modularity in metabolic networks. *Science* 297, 5586 (2002), 1551–1555. [\[41\]](#)
 - [128] READ, R. C., AND CORNEIL, D. G. The graph isomorphism disease. *Journal of Graph Theory* 1, 4 (1977), 339–363. [\[15\]](#)
 - [129] REICHL, L. *A modern course in statistical physics*, 2nd ed. Wiley, 1998. [\[78\]](#)
 - [130] SAARI, D. G. *Chaotic Elections! - A Mathematician Looks at Voting*. American Mathematical Society, 2001. [\[55\]](#)
 - [131] SAERENS, M., ACHBANY, Y., FOUSS, F., AND YEN, L. Randomized shortest-path problems: Two related models. *Neural Computation* 21 (2009), 2363–2404. [\[72\]](#), [\[77\]](#)
 - [132] SAERENS, M., FOUSS, F., YEN, L., AND DUPONT, P. The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning, ECML '04* (2004), 371–383. [\[45\]](#)
 - [133] SAIGO, H., NOWOZIN, S., KADOWAKI, T., KUDO, T., AND TSUDA, K. gBoost: a mathematical programming approach to graph classification and regression. *Mach. Learn.* 75, 1 (Apr. 2009), 69–89. [\[17\]](#)
 - [134] SAMPSON, S. A novitiate in a period of change. an experimental and case study of social relationships. In *Computational Intelligence, Theory and Applications* (1968). [\[119\]](#)
 - [135] SANFELIU, A., AND FU, K.-S. A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, 3 (1983), 353–362. [\[15\]](#)
 - [136] SCHAEFFER, E. Graph clustering. *Computer Science Review*, 1 (2007), 27–64. [\[3\]](#)
 - [137] SCHENKER, A., LAST, M., BUNKE, H., AND KANDEL, A. Classification of web documents using a graph model. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* (2003), IEEE, pp. 240–244. [\[2\]](#)

- [138] SCHOLKOPF, B., SMOLA, A., AND MULLER, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* 5, 10 (1998), 1299–1319. [\[112\]](#)
- [139] SCHOTT, J. *Matrix analysis for statistics*. Wiley, 2005. [\[79\]](#)
- [140] SCHRODINGER, E. *Statistical thermodynamics, 2nd ed.* Cambridge University Press, 1952. [\[78\]](#)
- [141] SEIDMAN, S., AND FOSTER, B. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6 (1978), 139–154. [\[74\]](#)
- [142] SENELLE, M., GARCIA-DIEZ, S., MANTRACH, A., SHIMBO, M., SAERENS, M., AND FOUSS, F. The sum-over-forests density index: identifying dense regions in a graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2013). [\[4\]](#)
- [143] SENELLE, M., SAERENS, M., AND FOUSS, F. Measures of similarity between graphs. *LSM Working Paper* (2013). [\[3\]](#)
- [144] SENELLE, M., SAERENS, M., AND FOUSS, F. The sum-over-forests clustering. *LSM Working Paper* (2013). [\[4\]](#)
- [145] SHAWE-TAYLOR, J., AND CRISTIANINI, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. [\[18\]](#)
- [146] SHERIDAN, R. P., AND KEARSLEY, S. K. Why do we need so many chemical similarity search methods? *Drug Discovery Today* 7, 17 (2002), 903 – 911. [\[12\]](#)
- [147] SHERVASHIDZE, N., SCHWEITZER, P., VAN LEEUWEN, E. J., MEHLHORN, K., AND BORGWARDT, K. M. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* 12 (2011), 2539–2561. [\[23\]](#), [\[26\]](#), [\[28\]](#), [\[35\]](#), [\[125\]](#)
- [148] SHERVASHIDZE, N., VISHWANATHAN, S. V. N., PETRI, T., MEHLHORN, K., AND BORGWARDT, K. M. Efficient graphlet kernels for large graph comparison. *Journal of Machine Learning Research - Proceedings Track 5* (2009), 488–495. [\[23\]](#)
- [149] SMITH, C. Drug target validation: Hitting the target. *Nature* 422, 6929 (2003), 341, 343, 345 passim. [\[13\]](#)
- [150] SMOLA, A. J., AND KONDOR, R. Kernels and regularization on graphs. In *Proceedings of the Conference on Learning Theory* (2003), M. Warmuth and B. Schölkopf, Eds., COLT '03, pp. 144–158. [\[44\]](#)
- [151] STREHL, A., GHOSH, J., AND CARDIE, C. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research* 3 (2002), 583–617. [\[121\]](#)

-
- [152] SUZUKI, I., HARA, K., SHIMBO, M., MATSUMOTO, Y., AND SAERENS, M. Investigating the effectiveness of Laplacian-based kernels in hub reduction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence* (2012), AAAI '12. [\[103\]](#)
- [153] TANG, J. M., AND SAAD, Y. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications* 19, 3 (2012), 485–501. [\[108\]](#)
- [154] TARJAN, R. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1, 2 (1972), 146–160. [\[118\]](#)
- [155] THOMA, M., CHENG, H., GRETTON, A., HAN, J., PETER KRIEGEL, H., SMOLA, A., SONG, L., YU, P. S., YAN, X., AND BORGWARDT, K. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the 9th SIAM International Conference on Data Mining* (2009), SDM '09. [\[17\]](#)
- [156] TONG, H., KOREN, Y., AND FALOUTSOS, C. Fast direction-aware proximity for graph mining. In *Proceedings of the ACM SIGKDD international conference on Knowledge Discovery and Data Mining* (2007), KDD '07, pp. 747–756. [\[45\]](#)
- [157] TUTTE, W. *Graph theory*. Cambridge University Press, 2002. [\[5\]](#), [\[72\]](#)
- [158] TVERSKY, A. Features of similarity. *Psychological Review* 84 (1977), 327–352. [\[41\]](#)
- [159] VAPNIK, V., AND LERNER, A. Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control* 24 (1963). [\[18\]](#)
- [160] VISHWANATHAN, S. V. N., SCHRAUDOLPH, N. N., KONDOR, R., AND BORGWARDT, K. M. Graph kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242. [\[22\]](#), [\[25\]](#)
- [161] VON LUXBURG, U., RADL, A., AND HEIN, M. Getting lost in space: Large sample analysis of the resistance distance. In *Advances in Neural Information Processing Systems (NIPS)* (2010), pp. 2622–2630. [\[43\]](#)
- [162] WAGNER, R. A., AND FISCHER, M. J. The string-to-string correction problem. *Journal of the ACM* 21, 1 (Jan. 1974), 168–173. [\[15\]](#)
- [163] WALE, N., WATSON, I. A., AND KARYPIS, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14, 3 (2008), 347–375. [\[32\]](#)
- [164] WALTERS, W. P., STAHL, M. T., AND MURCKO, M. A. Virtual screening—an overview. *Drug Discovery Today* 3, 4 (1998), 160–178. [\[13\]](#)

- [165] WANG, F., YUAN, C., XU, X., AND VAN BEEK, P. Supervised and semi-supervised online boosting tree for industrial machine vision application. In *Proceedings of the 5th International Workshop on Knowledge Discovery from Sensor Data* (2011), SensorKDD '11, ACM, pp. 43–51. [\[3\]](#)
- [166] WANG, M., HUA, X.-S., TANG, J., AND HONG, R. Beyond distance measurement: Constructing neighborhood similarity for video annotation. *IEEE Transactions on Multimedia* 11, 3 (2009), 465–476. [\[83\]](#)
- [167] WARD, J. H. Hierarchical grouping to optimize an objective function. *Journal of American Statistical Association* 58 (1963), 236–244. [\[120\]](#)
- [168] WASSERMAN, S., AND FAUST, K. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994. [\[73\]](#)
- [169] WEBB, A., AND COPSEY, K. *Statistical pattern recognition, 3rd ed.* John Wiley & Sons, 2011. [\[83\]](#)
- [170] WIENER, H. Structural determination of paraffin boiling points. *Journal of the American Chemical Society* 69, 1 (1947), 17–20. [\[16\]](#), [\[146\]](#)
- [171] XU, J., AND HAGLER, A. Chemoinformatics and drug discovery. *Molecules* 7, 8 (2002), 566–600. [\[13\]](#), [\[14\]](#)
- [172] YAN, X., AND HAN, J. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (2002), ICDM '02, IEEE Computer Society, pp. 721–. [\[17\]](#)
- [173] YEN, L., FOUSS, F., DECAESTECKER, C., FRANCO, P., AND SAERENS, M. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data and Knowledge Engineering* 68 (2009), 338–361. [\[52\]](#), [\[82\]](#), [\[83\]](#), [\[109\]](#), [\[111\]](#), [\[120\]](#)
- [174] YEN, L., MANTRACH, A., SHIMBO, M., AND SAERENS, M. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining* (2008), KDD '08, pp. 785–793. [\[29\]](#), [\[35\]](#), [\[47\]](#), [\[72\]](#)
- [175] YEN, L., SAERENS, M., AND FOUSS, F. A link analysis extension of correspondence analysis for mining relational databases. *IEEE Transactions on Knowledge and Data Engineering* 23, 4 (2011), 481–495. [\[84\]](#)
- [176] ZACHARY, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33 (1977), 452–473. [\[119\]](#)

- [177] ZHOU, D., BOUSQUET, O., LAL, T. N., WESTON, J., AND SCHÖLKOPF, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16* (2004), MIT Press, pp. 321–328. [\[45\]](#)
- [178] ZHOU, D., HUANG, J., AND SCHOLKOPF, B. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning* (2005), ICML '05, pp. 1041–1048. [\[53\]](#)
- [179] ZHOU, T., LÜ, L., AND ZHANG, Y. Predicting missing links via local information. *The European Physical Journal B-Condensed Matter and Complex Systems* 71, 4 (2009), 623–630. [\[41\]](#)
- [180] ZHU, X. Semi-supervised learning literature survey. Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005. [\[52\]](#)
- [181] ZHU, X., AND GOLDBERG, A. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2009. [\[52\]](#)

Extended table of contents

Table of contents	v
List of figures	ix
List of tables	xiii
General Introduction	1
Outline of the thesis	1
Managerial implications	2
Contributions	3
1 Preliminaries	5
I Similarity measures on graphs	9
2 Similarity measures between graphs	11
2.1 Introduction	12
2.1.1 Overview of the research topic	12
2.1.2 Applications	13
2.2 Review of graph comparison methods	15
2.2.1 Graph isomorphism	15
2.2.2 Graph-Edit distance	15

2.2.3	Graph features	16
2.2.3.1	The Wiener index [170]	16
2.2.3.2	The Morgan index [111]	16
2.2.3.3	The Zagreb index [59]	16
2.2.3.4	The Randic index [126]	17
2.2.4	Frequent subgraphs	17
2.2.5	Graph kernels	17
2.2.5.1	Classification and Support Vector Machines	18
2.2.5.2	Review of graph kernels	22
2.2.5.3	Marginalized kernel	23
2.2.5.4	Shortest-Path kernel	25
2.2.5.5	Weisfeiler-Lehman kernel	26
2.3	Proposed kernels	28
2.3.1	Modified Weisfeiler-Lehmann kernel	28
2.3.2	Randomized Shortest-Path kernel	28
2.3.3	Graph Features kernel	29
2.4	Experiments	32
2.4.1	Datasets	32
2.4.2	Experimental setup	33
2.4.3	Results	33
2.5	Chapter conclusion	35
3	Bag-of-Forests similarities between nodes of a graph	37
3.1	Review of similarity measures on nodes of a graph	38
3.1.1	Local similarity measures	38
3.1.1.1	Common Neighbors	38

3.1.1.2	Cosine coefficient	40
3.1.1.3	Jaccard index	40
3.1.1.4	Dice coefficient	41
3.1.1.5	Tversky index	41
3.1.1.6	Hub Promoted and Hub Depressed indices	41
3.1.1.7	Preferential attachment index	41
3.1.2	Global similarity measures	42
3.1.2.1	Katz index	42
3.1.2.2	The commute-time distance	42
3.1.2.3	The SimRank similarity measure	43
3.1.3	Kernel-based similarity measures	43
3.1.3.1	The exponential diffusion kernel	44
3.1.3.2	The Laplacian exponential diffusion kernel	44
3.1.3.3	The regularized Laplacian kernel	44
3.1.3.4	The von Neumann kernel	44
3.1.3.5	The commute-time kernel	45
3.1.3.6	The Markov diffusion kernel	46
3.1.4	Dissimilarities interpolating between commute-time and shortest-path	46
3.1.4.1	Logarithmically transformed forest similarities	46
3.1.4.2	The bag of paths framework	46
3.1.4.3	The randomized shortest-path dissimilarity	47
3.2	Introduction to the Matrix-Forest Theorem	47
3.3	A new forest similarity measure	49
3.3.1	The BoF potential distance	51
3.3.2	The BoF logarithmic distance	51

3.4	Experiments	52
3.4.1	Datasets	52
3.4.2	Experimental setup	53
3.4.3	Results	54
3.5	Chapter conclusion	68
II	Density measures on graphs	69
4	The Sum-over-Forests density index	71
4.1	Review of density measures on graphs	73
4.1.1	Dense subgraphs	73
4.1.2	Local density scores	75
4.1.2.1	Strength	75
4.1.2.2	Clustering Coefficient	75
4.1.3	Smoothing of local measures	76
4.1.3.1	Smoothing through Laplacian regularization	76
4.1.3.2	PageRank-like smoothing	76
4.1.4	Global density scores	77
4.2	The SoF density index	77
4.3	Computation of the partition function $\mathcal{Z}$	78
4.4	Computation of the SoF density index	79
4.5	Experiments	80
4.5.1	Datasets	81
4.5.2	Graph construction	83
4.5.3	Evaluation methods	84
4.5.4	Results	85

4.5.4.1	Correlation results	85
4.5.4.2	Visual results	94
4.5.4.3	k -Core results	97
4.6	Chapter conclusion	108
5	The Sum-over-Forests clustering on graphs	109
5.1	Review of density-based clustering	109
5.2	The SoF divisive clustering	112
5.2.1	Experiments	113
5.2.1.1	Datasets	113
5.2.1.2	Evaluation methods	113
5.2.2	Results	113
5.3	The SoF mode-seeking clustering	114
5.3.1	Experiments	119
5.3.1.1	Datasets	119
5.3.1.2	Evaluation methods	119
5.3.2	Results	121
5.4	Chapter conclusion	122
	General conclusion	125
	Main research results	125
	Applications	127
	Future prospects	128
	Bibliography	129
	Extended table of contents	145

