UNIVERSITY OF ROME "LA SAPIENZA"

SAPIENZA
UNIVERSITÀ DI ROMA

DOCTORAL THESIS

Pattern Recognition Techniques for Modelling Complex Systems in Non-Metric Domains

Author:
Alessio MARTINO

Supervisor:
Prof. Antonello RIZZI

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Computational Intelligence and Pervasive Systems Laboratory
Department of Information Engineering, Electronics and Telecommunications

February 21, 2020

Pattern Recognition Techniques for Modelling Complex Systems in Non-Metric Domains

PhD Thesis – University of Rome "La Sapienza"

PhD Programme: Information and Communication Technologies (XXXII cycle)

Version 1.2. Thesis defended on February 18th, 2020.

Examiners:

- Prof. Guglielmo D'Inzeo (University of Rome "La Sapienza")
- Prof. Luca Facheris (University of Florence)
- Dr. Andrea Bartolini (University of Bologna)
- Prof. Andrea Detti (University of Rome "Tor Vergata")
- Prof. Paolo Baccarelli (University of Rome "Roma Tre")

Referees:

- Prof. Luisa Di Paola (Università Campus Bio-Medico di Roma)
- Prof. Lorenzo Livi (University of Manitoba)

Declaration of Authorship

I, Alessio MARTINO, declare that this thesis titled, "Pattern Recognition Techniques for Modelling Complex Systems in Non-Metric Domains" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF ROME "LA SAPIENZA"

Abstract

Faculty of Information Engineering, Informatics, and Statistics
Department of Information Engineering, Electronics and Telecommunications

Doctor of Philosophy

Pattern Recognition Techniques for Modelling Complex Systems in Non-Metric Domains

by Alessio MARTINO

Pattern recognition and machine learning problems are often conceived to work on metric vector spaces, where patterns are described by multi-dimensional feature vectors. However, many real-world complex systems are more conveniently modelled by complex data structures such as graphs, which are able to capture topological and semantic information amongst entities.

This Thesis helps in bridging the gap between pattern recognition and graphs, with major emphasis on the hypergraphs domain. Six different strategies for solving graph-based pattern recognition problems are proposed, spanning several paradigms including kernel methods, embedding spaces and feature generation. The first two techniques map a graph towards a vector space by means of the spectral density of the Laplacian matrix and by means of topological invariants called the Betti numbers, respectively. Two additional techniques, according to the Granular Computing paradigm, map a graph towards a vector space by means of symbolic histograms. In a first case, simplices extracted from the simplicial complexes evaluated over the underlying graph are considered as candidate pivotal substructures for synthesising the symbolic histograms; in a second case, each path along a graph can be assigned a score that consider its specificity and sensitivity with respect to one of the problem-related classes and its inclusion in the candidate pivotal substructures is strictly related to its score. The final two techniques fall under the kernel methods umbrella: the first defines novel hypergraph kernels on the top of the simplicial complexes, the latter embraces a multiple kernel paradigm to exploit multiple graph representations simultaneously.

These techniques are tested on real-world problems related to three biological case studies, namely the solubility prediction and enzymatic properties discrimination in protein networks and the analysis of metabolic networks. Further, the most cutting-edge techniques are also tested on well-known benchmark datasets for graph classification and compared against current approaches in graph-based pattern recognition.

Contents

Declaration of Authorship iii

Abstract v

List of Figures ix

List of Tables xi

List of Abbreviations xiii

1 Introduction 1
1.1 Complex Systems and Complex Networks . . . 1
1.2 Computational Intelligence and Modelling Complex Systems . . . 3
1.3 Computational Biology Case Studies . . . 4
1.4 Thesis Scope and Outline . . . 6

2 Fundamentals of Graph Theory 9
2.1 Preliminary Definitions . . . 9
2.1.1 A Primer on Complex Networks . . . 13
2.2 Topological Data Analysis . . . 14

3 Pattern Recognition in Structured Domains 19
3.1 Preliminary Definitions . . . 19
3.2 Designing a Machine Learning System . . . 20
3.3 Mainstream Approaches . . . 22
3.3.1 Feature Generation / Feature Engineering . . . 24
3.3.2 Custom Dissimilarities in the Input Space . . . 24
3.3.3 Embedding Techniques . . . 26
Dissimilarity Space . . . 26
Information Granulation . . . 27
Kernel Methods . . . 28

4 Proposed Pattern Recognition Systems 31
4.1 Graph Classification by Spectral Density Estimation . . . 31
4.2 Graph Classification using the Betti Numbers Sequence . . . 32
4.3 Embedding over Simplicial Complexes . . . 34
4.4 Embedding via INDVAL . . . 37
4.5 Hypergraph Kernels . . . 39
4.6 Graph Classification using a Multiple Kernel Approach . . . 42
4.7 Final Remarks . . . 44

5 Tests and Results **47**
5.1 Datasets Description . . . 47
5.1.1 *E. coli* str. K12 PCN-EC . . . 47
5.1.2 *E. coli* str. K12 PCN-SOL . . . 49
5.1.3 Metabolic Networks . . . 50
5.1.4 Benchmark Datasets . . . 51
5.2 PCN Experiments: Enzymatic Properties . . . 51
5.2.1 EC Classification via Spectral Density . . . 51
5.2.2 EC Classification via Betti Numbers . . . 55
5.2.3 EC Classification by Embedding over Simplicial Complexes . . . 58
5.2.4 EC Classification using Hypergraph Kernels . . . 62
5.2.5 EC Classification using Multiple Kernel Machines . . . 62
5.2.6 Final Remarks . . . 69
5.3 PCN Experiments: Solubility Degree . . . 73
5.3.1 Solubility Prediction via Spectral Density . . . 73
5.3.2 Solubility Prediction via Betti Numbers . . . 74
5.3.3 Solubility Prediction via Embedding over Simplicial Complexes 75
5.3.4 Solubility Classification via Embedding over Simplicial Complexes . . . 76
5.3.5 Final Remarks . . . 78
5.4 Metabolic Pathways Experiments . . . 80
5.4.1 A Preliminary Investigation on the Gut Microbiota . . . 80
5.4.2 Metabolic Networks classification via INDVAL score . . . 87
On the Impact of the Threshold T . . . 91
5.4.3 Statistical Assessment of Classification Results . . . 93
5.4.4 Final Remarks . . . 97
5.5 Tests on Benchmark Datasets . . . 102
5.5.1 Hypergraph Kernels . . . 103
5.5.2 Embedding over Simplicial Complexes . . . 107
5.5.3 Embedding via INDVAL . . . 109

6 Other Research Activities **115**
6.1 Evolutive Agent-Based Clustering . . . 115
6.2 Distributed k-medoids Clustering . . . 116
6.3 Energy Management System Synthesis by ANFIS Networks . . . 121

7 Conclusions **125**

A Knowledge Discovery Data **127**
A.1 Multiple Kernel Machines (EC number classification) . . . 127
A.2 Embedding Simplicial Complexes (EC number classification) . . . 130
A.3 Embedding Simplicial Complexes (solubility classification) . . . 132

B Notes on Parallel and Distributed Evaluations **133**
B.1 Feature Generation using Graphs Spectral Density . . . 133
B.2 Feature Generation using the Betti Numbers . . . 134
B.3 Embedding over Simplicial Complexes . . . 134
B.4 Embedding via INDVAL . . . 137

Bibliography **141**

List of Figures

1.1 *E. coli* str. K12 RNA-binding protein (PDB 1HNR) 5
1.2 Krebs cycle from the *H. sapiens* metabolic pathway (KEGG M00009) . . 7

2.1 An example of undirected graph, its adjacency matrix and degree matrix 10
2.2 Comparison amongst Random, Scale-free and Small-world graphs . . 15
2.3 Example of Simplicial Complexes and their Homology 18

3.1 A (simplified) pattern recognition system workflow 20

4.1 Pre-processing chain for evaluating graphs spectral density 33
4.2 Topological configuration of a sample PCN (PDB 1HNR) at different scales 35
4.3 Barcode for sample PCN (PDB 1HNR) 36

5.1 Statistics for the initial 6685 PCNs dump (*E. coli* str. K12) 48
5.2 Statistics for PCN-EC 49
5.3 Statistics for PCN-SOL 50
5.4 Classes Distribution for the four Metabolic Network Datasets 52
5.5 EC classification via spectral density, weights vector for discriminating EC2 and EC4 55
5.6 EC classification via Betti numbers, vector space using the Clique Complex 56
5.7 EC classification via Betti numbers, weights heatmap 58
5.8 EC classification via multiple kernel machines, weights heatmap ($\alpha = 1$) 66
5.9 EC classification via multiple kernel machines, weights heatmap ($\alpha = 0.5$) 66
5.10 Schematic of the OCC_System and its learning procedure 67
5.10 ROC curves for EC number classification 71
5.11 Solubility prediction via spectral density, weights vector 75
5.12 Solubility prediction via Betti numbers, weights vector 75
5.13 Solubility classification via embedding over simplicial complexes, average performances as function of τ ($\alpha = 1$) 77
5.14 Solubility classification via embedding over simplicial complexes, average performances as function of τ ($\alpha = 0.5$) 78
5.15 F-statistic for metabolic pathways data (Problem 1) 81
5.16 F-statistic for metabolic pathways data (Problem 2) 81
5.17 F-statistic for metabolic pathways data (Problem 3) 82
5.18 F-statistic for metabolic pathways data (Problem 4) 82
5.19 Gut flora organisms clustering, the elbow plot 84
5.19 Gut flora organisms clustering, clusters composition 86
5.20 Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 1) 93

5.21 Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 2) 93
5.22 Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 3) 94
5.23 Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 4) 94
5.24 Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 1) 95
5.25 Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 2) 95
5.26 Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 3) 96
5.27 Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 4) 96
5.28 Performance of the blind classifier for all four metabolic pathways problems 97
5.29 (Hyper)graph kernels comparison, average accuracy on the test set . . 105
5.30 (Hyper)graph kernels comparison, average running times on the entire dataset 106
5.31 Edit-based hypergraph kernels, negative eigenfraction 107
5.32 Embedding over simplicial complexes performances on benchmark data, average accuracy on the Test Set 109
5.32 Embedding via INDVAL (path length 1) performances on benchmark data 111
5.32 Embedding via INDVAL (path length 2) performances on benchmark data 112
5.32 Embedding via INDVAL (path length 3) performances on benchmark data 113

6.1 MapReduce schema for the word-count problem 117

List of Tables

3.1 Some popular non-linear kernels for vector data 29

5.1 The six big EC nomenclature groups 47
5.2 Benchmark Datasets for Graph Classification 53
5.3 EC classification via spectral density, performances on the Test Set . . . 55
5.4 EC classification via Betti numbers (Clique Complex), performances on the Test Set 57
5.5 EC classification via Betti numbers (Vietoris-Rips Complex), performances on the Test Set 57
5.6 EC classification via embedding over simplicial complexes, performances on the Test Set at $\alpha = 1$. 61
5.7 EC classification via embedding over simplicial complexes, performances on the Test Set at $\alpha = 0.5$ 61
5.8 EC classification via Hypergraph Kernels, performance on the Test Set for Histogram Kernel 63
5.9 EC classification via Hypergraph Kernels, performance on the Test Set for Jaccard Kernel 63
5.10 EC classification via Hypergraph Kernels, performance on the Test Set for Edit Kernel 63
5.11 EC classification via Hypergraph Kernels, performance on the Test Set for Stratified Edit Kernel 63
5.12 EC classification via multiple kernel machines, performances on the Test Set 65
5.13 EC classification: comparison between OCC and MK, performances on the Test Set 69
5.14 Variance explained and statistical significance for the seven models . . 72
5.15 Hydrophilicity contribution to score for different classes 73
5.16 Polarity contribution to score for different classes 73
5.17 Solubility prediction via spectral density, performances on the Test Set 74
5.18 Solubility prediction via Betti numbers, performances on the Test Set . 75
5.19 Solubility prediction via embedding over simplicial complexes, performances on the Test Set 76
5.20 Pearson correlation coefficients between polarity and hydrophilicity . 79
5.21 Gut flora organisms clustering, chemical reaction INDVAL scores . . . 86
5.22 Metabolic Networks classification, Problem 1 with K-NN, average performances on the Test Set 87
5.23 Metabolic Networks classification, Problem 2 with K-NN, average performances on the Test Set 87
5.24 Metabolic Networks classification, Problem 3 with K-NN, average performances on the Test Set 88
5.25 Metabolic Networks classification, Problem 4 with K-NN, average performances on the Test Set 88

5.26 Metabolic Networks classification, average size of the starting alphabet for building the INDVAL-based embedding space ($T = 50$) 89
5.27 Metabolic Networks classification, Problem 1 via INDVAL-based embedding, average performances on the Test Set ($T = 50$) 89
5.28 Metabolic Networks classification, Problem 2 via INDVAL-based embedding, average performances on the Test Set ($T = 50$) 90
5.29 Metabolic Networks classification, Problem 3 via INDVAL-based embedding, average performances on the Test Set ($T = 50$) 90
5.30 Metabolic Networks classification, Problem 4 via INDVAL-based embedding, average performances on the Test Set ($T = 50$) 91
5.31 Metabolic Networks classification, Problem 4 via INDVAL-based embedding, average performances on the Test Set ($T = 30$) 92
5.32 Metabolic Networks classification, relevant edges for Problem 1 ($T = 50$) 102
5.33 Metabolic Networks classification, relevant edges for Problem 2 ($T = 50$) 103
5.34 Metabolic Networks classification, relevant edges for Problem 3, class 2 only ($T = 50$) 103
5.35 Kernel parameters to be tuned 104

A.1 Selected proteins in order to discriminate EC 1 (Oxidoreductases) vs. all the rest 127
A.2 Selected proteins in order to discriminate EC 2 (Transferases) vs. all the rest 127
A.3 Selected proteins in order to discriminate EC 3 (Hydrolases) vs. all the rest 128
A.4 Selected proteins in order to discriminate EC 4 (Lyases) vs. all the rest 128
A.5 Selected proteins in order to discriminate EC 5 (Isomerases) vs. all the rest 129
A.6 Selected proteins in order to discriminate EC 6 (Ligases) vs. all the rest 129
A.7 Selected proteins in order to discriminate not-enzymes vs. all the rest . 130
A.8 Amino-acids nomenclature table 130
A.9 Selected simplices for EC number classification 131
A.10 Selected simplices for solubility classification 132

List of Abbreviations

avg	**a**verage
e.g.	**e**xempli **g**ratia (transl. *for example*)
i.e.	**i**d **e**st (transl. *that is to say*)
std	**s**tandard **d**eviation
ACC	**Acc**uracy
ANFIS	**A**daptive **N**euro**F**uzzy **I**nference **S**ystem
API	**A**pplication **P**rogram **I**nterface
AUC	**A**rea **U**nder the (ROC) **C**urve
BSAS	**B**asic **S**equential **A**lgorithmic **S**cheme
DNA	**D**eoxyribo**N**ucleic **A**cid
E-ABC	**E**volutive **A**gent **B**ased **C**lustering
EC	**E**nzyme **C**ommission
EK	**E**dit **K**ernel
EQ	**E**stimation **Q**uality
FN	**F**alse **N**egatives
FP	**F**alse **P**ositives
GrC	**Gr**anular **C**omputing
GPU	**G**raphics **P**rocessing **U**nit
GRALG	**GR**anular Computing **A**pproach for **L**abeled **G**raphs
HK	**H**istogram **K**ernel
IP	**I**dentified **P**atterns
K-**NN**	*K*-**N**earest **N**eighbours
KEGG	**K**yoto **E**ncyclopedia of **G**enes and **G**enomes
LD-ABCD	**L**ocal **D**issimilarities-**A**gent-**B**ased **C**lusters **D**iscoverer
MinSoD	(element that leads to the) **Min**imum **S**um **o**f **D**istances (syn. *medoid*)
MKMD	**M**ultiple **K**ernels over **M**ultiple **D**issimilarities
MLP	**M**ulti**L**ayer **P**erceptron
MSE	**M**ulti**S**quared **E**rror
NEF	**N**egative **E**igen**F**raction
NPV	**N**egative **P**redictive **V**alue
OCC	**O**ne **C**lass **C**lassification System (also, OCC_System)
PCN	**P**rotein **C**ontact **N**etwork
PPV	**P**ositive **P**redictive **V**alue
PDB	**P**rotein **D**ata **B**ank
PK	**P**ropagation **K**ernel
RDD	**R**esilient **D**istributed **D**ataset
RNA	**R**ibo**N**ucleic **A**cid
ROC	**R**eceiver **O**perating **C**haracteristic
RW	**R**andom **W**alk (Kernel)
SEK	**S**tratified **E**dit **K**ernel
SNS	**Sns**itivity
SPC	**Spc**ificity
SVM	**S**upport **V**ector **M**achine

SVR	**S**upport **V**ector **R**egression
TDA	**T**opological **D**ata **A**nalysis
TN	**T**rue **N**egatives
TP	**T**rue **P**ositives
WCSoD	**W**ithin-**C**lusters **S**um-**o**f-**D**istances
WJK	**W**eighted **J**accard **K**ernel
WL	**W**eisfeiler–**L**ehman (Subtree Kernel)
WLSP	**W**eisfeiler–**L**ehman **S**hortest **P**ath (Kernel)

Chapter 1

Introduction

1.1 Complex Systems and Complex Networks

Complex systems are everywhere in nature: they are by far more frequent than 'simple' ones, which are the true outliers in our world.

However, there is still some debate about a precise and rigorous definition of *complex system*. This is mainly due to the fact that complex systems are nowadays faced by several disciplines (including, but not limited to, biology, engineering, physics, chemistry), each one bringing its own concepts, definitions and viewpoints to the subject matter. The lack of a shared and precise definition about complex systems has been remarked in the seminal paper "From Complexity to Perplexity", written by John Horgan in 1995 [170]. Notwithstanding that, most authors agree that a complex system should exhibit at least some of the following behavioural facets:

- abrupt changes of macroscopic behaviour, which may as well be chaotic
- scale invariance
- self-organisation, with possibly hierarchical structure
- power law-type relationships.

By observing these kind of systems, the following structural peculiarities emerge:

- the system is composed by many mutually interactive elements
- elements' behaviour is characterised by non-linear dynamics
- the network representing the causal relationships amongst elements contains loops.

Such 'interactive elements' can be interpreted as useful and meaningful entities to be chosen according to the desired representation (description) of the system. For example, proteins can be considered as atomic entities in the network of chemical reactions in a biological cell (the widely-known metabolic pathways), but if the system to be described is the protein itself, then its secondary structures or (at deeper level) its amino-acids can be considered as atomic entities. Similarly, neurones or (at larger scale) individual parts of the brain (e.g., cortices) can be considered as the atomic entities when modelling the brain as a network.

These examples of complex systems underline a frequent property, concerning the fact the usually complexity arises in the form of a hierarchical organisation, as nested system-of-systems. From this last point of view, it is possible to consider causal relations between elements belonging to different levels in the hierarchical organisation. When the network of these relations contains a loop, sometimes it is

referred to as 'strange loop'; that is, a causal loop between different levels of the hierarchy [167]. This property is strongly related with the emergence of the most interesting behaviours of a given system-of-systems when considered as a whole.

In 1948, Warren Weaver in his seminal work [377] proposed a three-fold partition of science styles. Specifically, scientific themes can be divided into:

Class 1: Problems of Simplicity

Class 2: Problems of Disorganised Complexity

Class 3: Problems of Organised Complexity

Weaver's class 1 corresponds to problems which can be solved in terms of differential and/or integro-differential equations and, at the same time, they allow a high degree of abstraction. For example, in order to study gravitational laws, a planet can be sketched as a pure dimensionless point whose only mass and distance from the Sun shall be considered.

Weaver's class 2 problems allow higher generalisation than class 1 problems and the reasoning behind the system description is quite different: instead of focusing on efficient abstract descriptions of the involved atomic elements, one shall consider only a coarse-grained knowledge, usually by means of statistics on a transfinite number of players. Thermodynamic quantities such as pressure, volume and temperature belong to this class. Thus, an immediate comparison between class 1 and class 2 problems arise: class 1 problems need few actors with stable interactions amongst them, class 2 problems need a lot of actors with negligible interactions.

Finally, Weaver's class 3 can be considered as the 'middle kingdom' of complexity (or 'the middle way', according to Robert B. Laughlin and colleagues [211]). In class 3 problems there are many actors (even if not-so-many as in class 2) with non-negligible interactions amongst them: here biological systems, one of the main focuses of this work, lie.

Graphs (or networks[1]) are the archetype of organised complexity: a set of nodes (atomic entities) are pairwise connected by mutual interactions (edges). The wiring architecture of these graphs can vary in both space and time and it is of utmost importance to get quantitative similarities and differences among them. When graphs are adopted to represent only topological information concerning a set of objects and their relations, the network approach can roughly be described as the answer to the following question [149, 150, 273]:

> *What can we derive from the sole knowledge of the wiring diagram of a system?*

The most crucial questions especially at the frontiers of biomedical sciences demands a reliable answer to the above question. Fields (just to name a few) that are increasing their formalisation in terms of network representations are: neuroscience at both clinical and basic research level [55, 312], biochemistry [35, 116, 384], cancer research [392], ecology [143] and structural biology [233], with protein science that still is the most explored field in the realm of organised complexity [15, 20, 62, 114, 298, 365, 390]. Nonetheless, life sciences are not the only fields in which complex networks play a huge modelling role: think about social networks [375], where people are connected by some kind of relationship; the World Wide Web, where webpages point to each other by hyperlinks [109, 381]; power grids, where electricity is distributed

[1] Albeit with *graph* one refers to an abstract mathematical object whereas a *network* is its physical, real counterpart, these terms will be used interchangeably throughout this work.

amongst different geographic areas [216, 217]; natural language processing [81, 96, 97]; computer vision and image processing [3, 13, 43, 44, 106, 166].

Especially in the first decade of the 21st century, seminal works by Albert-László Barabási, Steven Henry Strogatz and others paved the way for graph-theoretical models for analysing complex systems [32, 34, 46, 92–94, 104, 114, 120, 150, 168, 227, 275, 289, 290, 330, 349, 350]. Indeed, by quoting Vincenzo Nicosia and colleagues [292]:

> *Networks are the fabric of complex systems.*

This new wave of graph-based methods allowed a rich set of measures which, in turn, provided a multifaceted description of complex networks [5, 122, 129, 142, 156, 165, 227, 290, 297, 319, 329].

1.2 Computational Intelligence and Modelling Complex Systems

Computational Intelligence, formerly known as *Soft Computing* thanks to pioneer Lotfi Aliasker Zadeh [397], is a set of data processing techniques tolerant to imprecisions, uncertainty, partial truth and approximation in data and/or models, aimed to provide robust and low-cost solutions and to achieve tractability when dealing with complex systems [251, 340]. Such toolbox includes mostly biologically-inspired algorithms, usually exploiting inductive reasoning (i.e., based on generative logic inferences, such as analogy and induction [66]). Basically, in this toolbox it is possible to find:

- artificial neural networks
- fuzzy logic and neuro-fuzzy systems
- evolutionary computation and derivative-free optimisation metaheuristics.

Computational intelligence with this impressive armamentarium of computational tools is usually employed to design powerful data-driven modelling systems. Being able to synthesise a predictive model of a given (physical or even abstract) process $\mathcal{P}$ is a fundamental topic in all natural sciences, as well as in engineering. Before the widespread of digital computing devices, modelling was performed 'by hand', mostly relying on field-experts (*analytical modelling*): the field-expert used to identify meaningful quantities and relations amongst them, with the ultimate goal of writing a set of integro-differential equations. Needless to say, this approach requires a clear understanding of the process to be modelled. However, when a meaningful sampling $\tilde{\mathcal{P}}$ of the target process $\mathcal{P}$ is available, a second approach (*data-driven modelling*) can be pursued: it consists in writing an algorithm[2] able to automatically synthesise a model M of $\mathcal{P}$ by exploiting $\tilde{\mathcal{P}}$ according to some predefined optimality criteria. Nowadays, this modelling approach is known as *machine learning*. A formal definition can be found in [276], where Tom Mitchell considers machine learning as the following, well-posed problem:

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

[2] Usually suitable to be run on a Von Neumann computing architecture.

More broadly, machine learning can be defined as a set of complex intelligent information processing systems, usually defined by means of adaptive learning algorithms, able to act without being explicitly programmed or (in order words) able to learn from data and experience.

1.3 Computational Biology Case Studies

In Section 1.1, major emphasis has been put to biological systems and the widespread use of graphs in biological sciences. This is not by chance. Indeed, by quoting Sergio Barbarossa[3]:

> *Biology is the science of networks.*

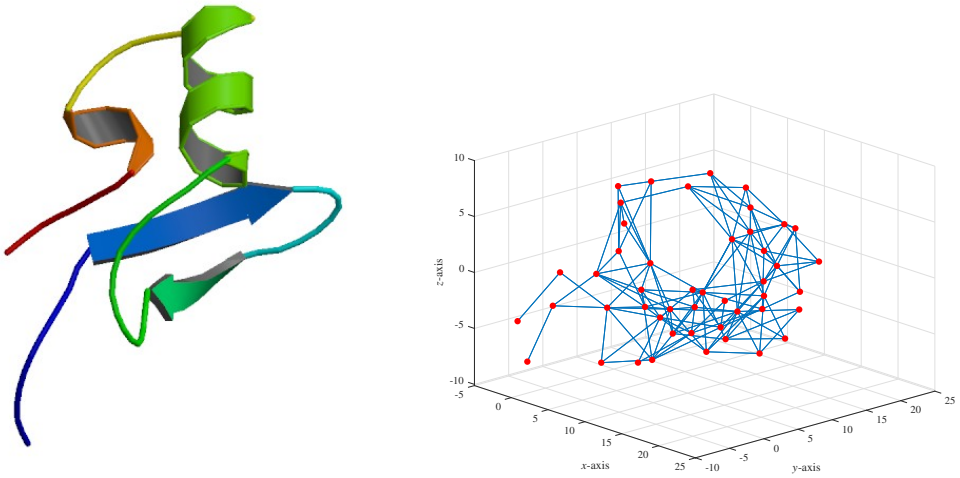
In order to show the effectiveness of the techniques presented in this work, two real-world biological systems will be analysed: Protein Contact Networks (PCNs) [114, 204] and metabolic pathways [139, 206].

A PCN is a minimalistic graph-based representation of the 3D folded state of a protein (see Figure 1.1 for a visual comparison between the well-known ribbon diagram[4] and the PCN). Proteins are macromolecules in charge of performing a plethora of functions within living organisms, including DNA replication [2], providing cytoskeleton [382], catalysing enzymatic reactions [36], cell signalling and intracellular transport [237]. Proteins biosynthesis is composed by multiple steps, the most important being *transcription* and *translation* [185]. The DNA sequence of a gene encodes the sequence of amino-acids that forms each protein. This DNA sequence undergoes transcription, hence becoming pre-messenger RNA. The RNA sequence, in turn, is loaded into the ribosome where it undergoes translation. The ribosome, thanks to the genetic code, performs translation: each triplet of nucleotides (codon) is translated into one of the (approximately) 20 amino-acids. As this amino-acid chain leaves the ribosome, the *primary structure* is formed and the process called *protein folding* starts [21, 383]. Due to hydrogen bonds, local sub-structures start emerging, leading to the so-called *secondary structure*, whether the chain does not fall into a random coil configuration. α-helices, β-sheets and turns are the three most common secondary structures. Finally, the spatial relationship between secondary structures determines the overall folded protein shape, namely the *tertiary structure* [2]. The peculiar three-dimensional shape (structure) of a protein is unambiguously determined by the primary structure (Anfinsen's dogma [7]) and is at the basis of its function. Furthermore, this 3D configuration goes into slightly but crucial changes in order to adapt to its micro-environment. Indeed, the deformation affects the interaction potentials between protein's atoms and the external environment, allowing it to carry out a specific function. There is a deep relation between the function and the structure of a protein, and investigating the latter is a fundamental step in understanding the former. A thorough comprehension of how a protein works is in turn of great significance for a variety of practical settings, like drug design and the diagnosis of diseases [94, 347, 348, 367, 370]. PCNs are (by definition) unweighted, undirected and unlabelled networks which are built starting from the native 3D structure [132, 390] where nodes correspond to amino-acids and edges exist between any two nodes whether they are in spatial proximity in the 3D protein

[3] Private communication.

[4] The *ribbon representation* (also *cartoon representation*) is a schematic 3D representation of the folded protein, highlighting the structural organisation of the backbone and the secondary structures: α-helices are shown as coiled ribbons, β-sheets as arrows and random coils as thick tubes or lines.

folded state, hence if their respective α-carbon atoms are arranged at a distance that is within a given threshold of typically $[4, 8]$Å [100, 114, 226, 227, 233, 234, 244, 249, 251, 262]. The lower bound is usually defined in order to discard trivial first-order contacts of adjacent residues along the backbone, whereas the upper bound is defined according to the peptide bonds geometry [304] (8Å roughly correspond to two Van der Waals radii [233]). Despite their 'minimalistic' definition[5], PCNs allow for a reliable reconstruction of the overall folded protein structure [363, 379], as well as an efficient description of the relevant biological properties, such as allosteric effect and identification of active sites [94, 115].



(A) Ribbon Representation (B) PCN Representation

FIGURE 1.1: *E. coli* str. K12 RNA-binding protein (PDB 1HNR).

Metabolic pathways are chains of chemical reactions occurring within a cell, where the products of one reaction are the substrates for the subsequent reactions. Such intermediates (be it reactants, substrate, products, and and like) are usually referred to as *metabolites*. Clearly, a metabolic pathway can be modelled by a network where nodes correspond to metabolites and edges are scored whether there exist a chemical reaction between the two nodes. Conversely to PCNs, metabolic networks might not be undirected: indeed, despite all chemical reactions are technically reversible, conditions in the cell are often such that thermodynamically reactions flow in one direction [83, 84]. The network formalism is a static representation of the chemical reactions that can happen in a particular organism: it derives from the knowledge of the reaction(s) catalysed by each single enzyme of the organism. The complete list of those enzymes is known with no uncertainty and derives from the total DNA sequencing of the organism. The involved players (i.e., metabolites, which are the nodes of the network) are physical objects that interact by a dynamical link (chemical reaction catalysed by an enzyme) but the kinetic data cannot be represented in a unique way (in addition they are very difficult to ascertain *in vivo*) because they can change up to four orders of magnitude depending to microenvironmental conditions (pH, nutrient concentration state, cell cycle phase and so on) thus inhibiting any consistent network representation. This is why in literature a binary yes/no information linked to the presence/absence of enzymes is currently

[5]It is worth remarking that the amino-acid type (node label) and the proper distance between amino-acids (edge label) are deliberately discarded in order to focus on proteins' topological structure.

used (i.e., presence/absence of a reaction transforming metabolite i into metabolite j) [299, 360]. The interruption of the global connectivity of the metabolic network caused by a mutation that impairs the functionality of the enzyme catalysing an essential chemical reaction (and the consequent inability to follow a given pathway) is a necessary (but not sufficient) condition for organism death [299]. The crucial role of metabolic networks makes them a preferred viewpoint for a lot of biomedical applications [48, 91] and practical settings, for example when it comes to detect comorbidity patterns (e.g., obesity vs. diabetes) in sick patients [212, 328]. Comorbidity can easily be driven by a malfunctioning in catalysing a given reaction: due to the chain-like nature of metabolic pathways, a single enzyme defect may affect other chemical reactions, leading to a potentially dangerous domino-effects. Figure 1.2 shows a very peculiar module on the human metabolic network: the citrate cycle (also widely-known as TCA cycle or Krebs cycle).

1.4 Thesis Scope and Outline

The aim of this work is the development of novel pattern recognition systems able to deal with structured domains, with particular emphasis on the (hyper)graphs domain. Graphs, by definition, lie in a non-geometric space [230, 232, 251], which is (computationally speaking) way more difficult than the plain Euclidean space in which all pattern recognition systems can be used without alterations.

In order to show the effectiveness of the proposed techniques, the two real-world biological systems briefly described in Section 1.3 will be analysed. As regards PCNs, this work aims at addressing whether it is possible to predict proteins' enzymatic class and solubility degree starting from the graph-based representation of their folded state. As regards metabolic networks, we ask whether it is possible to automatically recognise metabolic pathways belonging to different organisms.

The most cutting-edge techniques amongst the proposed ones will be also tested on freely-available benchmark datasets and compared with state-of-the-art graph-based classification systems.

This thesis is organised as follows: Chapters 2 and 3 will provide the reader some theoretical background. Specifically, Chapter 2 regards graph theory: starting from basic definitions, the most important matrix representations and their peculiar properties in terms of network characterisation are introduced. Then, the discussion moves towards the novel Topological Data Analysis field by introducing hypergraphs and simplicial complexes. Conversely, Chapter 3 extends the computational intelligence part, briefly introduced in Section 1.2. Starting from basic definitions about supervised and unsupervised learning, the Chapter will mainly discuss both challenges and strategies when dealing with pattern recognition in structured domains, namely domains which (conversely to the standard Euclidean space) might not be metric in nature.

Chapter 4 describes the proposed six graph-based pattern recognition systems, highlighting their respective strengths and weaknesses, whereas Chapter 5 presents the computational results on both real-world data (PCNs and metabolic pathways) and benchmark data.

Chapter 6 describes some major research activities that somehow shift from the whole graph-based pattern recognition dissertation.

Finally, Chapter 7 concludes this work, remarking future research endeavours.

This thesis also features two appendices: Appendix A contains in-depth results about the knowledge discovery carried out a-posteriori over some of the proposed

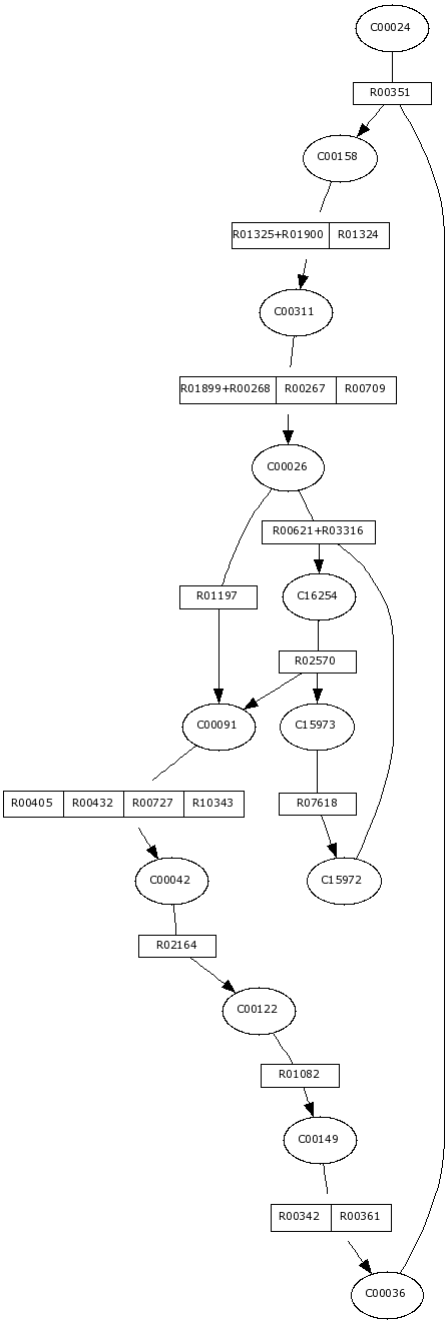


FIGURE 1.2: Krebs cycle from the *H. sapiens* metabolic pathway (KEGG M00009).

technique, whereas Appendix B sketches parallel and distributed implementations for some of the proposed techniques in order to face computationally expensive problems and/or large datasets.

Chapter 2

Fundamentals of Graph Theory

2.1 Preliminary Definitions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V}$ is the finite set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. For short, let $n = |\mathcal{V}|$.

A graph is said to be *undirected* if edges have no orientation: if an edge exists between any two nodes, say $(u, v) \in \mathcal{E}$ for $u, v \in \mathcal{V}$, then (v, u) is identical to (u, v). If edges have orientation, that is $(v, u) \in \mathcal{E}$, but (u, v) might not exist in $\mathcal{E}$, then the graph is said to be *directed*. Further, a graph is said to be *simple* if there are no self-loops (i.e., no nodes connected to themselves) and there are no multi-edges[6] (i.e., pair of nodes connected by more than one edge).

Clearly, graphs are able to capture topological information from the entities involved. However, nodes and/or edges can be equipped with semantic information (in terms of more or less complex attributes): in this case one shall refer to as *labelled graphs*. The most simple type of labelled graph is widely-known as *weighted graph*, where nodes have no attributes and edge attributes are plain real-valued scalars denoting the 'strength' of the connection between the two nodes at their extremities.

The most straightforward and complete way to represent a graph is by means of its *adjacency matrix* $\mathbf{A}$, a square $n \times n$ matrix defined as

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Trivially, for simple and unweighted graphs, one shall expect a binary and zero-diagonal matrix. If the graph is undirected, then the adjacency matrix is also symmetric.

Starting from the adjacency matrix it is possible to evaluate the *degree matrix* $\mathbf{D}$, a diagonal $n \times n$ matrix defined as[7]

$$\mathbf{D}_{i,j} = \begin{cases} D(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where in turn $D(v_i)$, the *degree* of node v_i, is defined as the number of nodes connected to it

$$D(v_i) = \sum_{j=1}^{n} \mathbf{A}_{i,j} \tag{2.3}$$

[6] Whether multi-edges exist, one shall refer to as *multi-graph*.

[7] This definition holds for undirected networks. If the network is directed one shall consider separately the *in-degree* and the *out-degree* of each node, namely the number of incoming and outgoing links, respectively.

Figure 2.1 shows an example of simple, undirected and unweighted graph, along with its corresponding adjacency and degree matrices.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

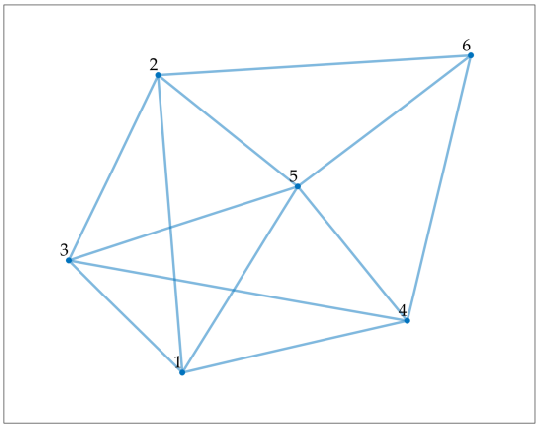


FIGURE 2.1: An example of undirected graph (bottom centre), its adjacency matrix (top left) and degree matrix (top right). It is noteworthy that the node numbers do not have to be intended as their respective attributes (labels); rather, they have been added for consistency with the row/column ordering in the adjacency and degree matrices.

Starting from $\mathbf{A}$ and $\mathbf{D}$, the *Laplacian matrix* $\mathbf{L}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.4}$$

The Laplacian matrix is the generalisation of the differential Laplacian operator on graphs [289] and provides many information about the topology: as such, it has been widely used in many facets of graph theory, including graph partitioning, study of dynamical network processes, network connectivity and the like [205, 209, 238, 308, 405]. The graph Laplacian is related to many local and global properties of the network [272, 278] and its spectrum provides a compact representation of the graph, which can be used for a better understanding of the network organisation [70, 277, 401, 406].

Let $\Lambda^{(A)} = \{\lambda_1^{(A)}, \ldots, \lambda_n^{(A)}\}$ and $\Lambda^{(L)} = \{\lambda_1^{(L)}, \ldots, \lambda_n^{(L)}\}$ be the set of eigenvalues of $\mathbf{A}$ and $\mathbf{L}$, respectively. It is possible to define the *graph energy* E and the *Laplacian energy* LE respectively as [162]:

$$E = \sum_{i=1}^{n} \left| \lambda_i^{(A)} \right| \tag{2.5}$$

$$LE = \sum_{i=1}^{n} \left| \lambda_i^{(L)} - \frac{2|\mathcal{E}|}{n} \right| \tag{2.6}$$

From Eqs. (2.1)–(2.4) it is possible to define the *normalised Laplacian matrix* $\overline{\mathbf{L}}$ as:

$$\overline{\mathbf{L}} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{2.7}$$

As per the un-normalised counterpart, the normalised Laplacian is another important matrix representation as it conveys many structural and dynamical properties [234, 265, 277]. Further, the normalised Laplacian presents some interesting algebraic properties [60, 75, 184], notably:

- it is positive semi-definite: there are no negative eigenvalues
- all its eigenvalues (its *spectrum*) lie in range $[0, 2]$, regardless of the underlying graph $\mathcal{G}$ (i.e. number of nodes, number of vertices and the like)
- the multiplicity of the null eigenvalue equals the number of connected components of $\mathcal{G}$: as such, there exist at least one null eigenvalue.

Let us consider the spectral decomposition of $\overline{\mathbf{L}}$:

$$\overline{\mathbf{L}} = \mathbf{V}\boldsymbol{\Lambda}^{(\overline{L})}\mathbf{V}^T \tag{2.8}$$

where $\boldsymbol{\Lambda}^{(\overline{L})} = \text{diag}\left\{\lambda_1^{(\overline{L})}, \ldots, \lambda_n^{(\overline{L})}\right\}$ is a diagonal matrix containing the eigenvalues in increasing order and $\mathbf{V}$ contains the corresponding unitary-length eigenvectors. The heat equation associated to $\overline{\mathbf{L}}$ reads as [202, 387, 388]

$$\frac{\partial \mathbf{H}(t)}{\partial t} = -\overline{\mathbf{L}}\mathbf{H}(t) \tag{2.9}$$

The solution to Eq. (2.9), $\mathbf{H}(t)$, is the *heat kernel matrix* at time t.

$$\mathbf{H}(t) = \exp\left\{-t\overline{\mathbf{L}}\right\} = \mathbf{V}\exp\left\{-t\boldsymbol{\Lambda}^{(\overline{L})}\right\}\mathbf{V}^T = \sum_{i=1}^{n} \exp\left\{-\lambda_i^{(\overline{L})}t\right\}\mathbf{v}_i\mathbf{v}_i^T \tag{2.10}$$

In short, the heat kernel operator simulates the heat diffusion across the network and, obviously, the diffusion process is driven by the structure of the network itself. For example, in a tightly-connected network, one shall expect a very fast diffusion process that reaches an equilibrium point after a relatively small time, whereas in a weakly-connected and highly-modular network, one shall expect a slower diffusion process. The heat kernel matrix in Eq. (2.10) is an $n \times n$ doubly-stochastic and time varying matrix which describes the amount of some diffusing substance across the edges of the network: indeed, the element $\mathbf{H}_{i,j}(t)$ scores the amount of substance that, starting from node i, goes towards node j after time t. The heat kernel matrix shows the following properties:

$$\mathbf{H}(t) \simeq \begin{cases} \mathbf{I} - \overline{\mathbf{L}} \cdot t & \text{if } t \to 0 \\ \exp\left\{\lambda_2^{(\overline{L})} \cdot t\right\}\mathbf{v}_2\mathbf{v}_2^T & \text{if } t \to \infty \end{cases} \tag{2.11}$$

where $\mathbf{I}$ is the identity matrix, $\lambda_2^{(\overline{L})}$ is the smallest non-zero eigenvalue of $\overline{\mathbf{L}}$ and $\mathbf{v}_2$ is the corresponding eigenvector (also commonly known as the Fiedler vector [75]). From Eq. (2.11) it is possible to see that the short-time diffusion behaviour is determined by the local structure of the network, whereas the large-time behaviour is driven by the global structure of the network[8].

[8] $\lambda_2^{(\overline{L})}$ is also known as the *spectral gap* and it can be seen as a measure of separateness of the graph into strongly connected components [6].

The *heat kernel trace* $HT(t)$ of $\mathbf{H}(t)$ is given by

$$HT(t) = \mathrm{Tr}\left\{\mathbf{H}(t)\right\} = \sum_{i=1}^{n} \exp\left\{-\lambda_i^{(\overline{L})} t\right\} \tag{2.12}$$

For a connected graph, at $t = 0$ we have $HT(t) = n$, whereas as $t \to \infty$ we have[9] $HT(t) \to 1$. The heat trace $HT(t)$ can be interpreted as the number of connected components under the diffusion viewpoint. For example, at $t = 0$, each node of the network generates some information which is not yet propagated towards other nodes, hence $HT(t = 0) = n$ (each node has its own information). As t grows, nodes start propagating their information farther and farther across the network until a steady-state is reached: that is, where all nodes are in contact with each other. Under the information viewpoint, this corresponds to a single connected component, hence $HT(t \to \infty) \to 1$.

The *heat content* $HC(t)$ of $\mathbf{H}(t)$ is given by

$$HC(t) = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \mathbf{H}_{i,j}(t) = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{k=1}^{n} \exp\left\{-\lambda_k^{(\overline{L})} t\right\} \mathbf{v}_k(v_i)\mathbf{v}_k(v_j) \tag{2.13}$$

where $\mathbf{v}_k(v_i)$ is the value related to node v_i in the k^{th} eigenvector.
The MacLaurin series for the negative exponential reads as

$$\exp\left\{-\lambda_k^{(\overline{\mathbf{L}})} t\right\} = \sum_{m=0}^{\infty} \frac{\left(-\lambda_k^{(\overline{L})} t\right)^m t^m}{m!} \tag{2.14}$$

and substituting Eq. (2.14) in Eq. (2.13) yields

$$HC(t) = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{k=1}^{n} \sum_{m=0}^{\infty} \frac{\left(-\lambda_k^{(\overline{L})} t\right)^m t^m}{m!} \mathbf{v}_k(v_i)\mathbf{v}_k(v_j) \tag{2.15}$$

By re-writing Eq. (2.13) in terms of power series as [264]

$$HC(t) = \sum_{m=0}^{\infty} q_m t^m \tag{2.16}$$

the set of coefficients q_m are the so-called *heat content invariants* and can be evaluated in closed-form as

$$q_m = \sum_{i=1}^{n} \left(\left(\sum_{v \in \mathcal{V}} \mathbf{v}_i(v) \right)^2 \right) \frac{\left(-\lambda_i^{(\overline{L})}\right)^m}{m!} \tag{2.17}$$

Conversely to the heat kernel trace, the heat content invariants rely on both eigenvalues and eigenvectors: this aspect makes them preferable over the plain heat kernel trace since the joint consideration of both eigenvalues and eigenvectors for graph characterisation helps in avoiding the co-spectrality problem [95, 127].

[9] Recall that if the graph is connected, there exist one connected component, hence only one null eigenvalue.

2.1.1 A Primer on Complex Networks

At this point one might ask what makes networks 'complex'. Complex networks show non-trivial topological features (i.e., connectivity patterns) that do not occur in 'simple networks' such as purely regular graphs (lattices) or purely random graphs. That is because complexity lies in-the-middle: for a medium number of interactions, one gets the maximum entropy [197].

The two main families of complex networks include the so-called *scale-free networks* [26] and *small-world networks* [4, 376]. Scale-free networks are characterised by a steady-growth process when new entities interact with already-existing entities. Scale-free networks trace back to late 90's [25], when the World Wide Web was in its early stage: Barabási and colleagues noticed that new (source) webpages used to link to existing (target) webpages with a probability driven by the degree of the target webpage. This phenomenon is also called as *preferential attachment* or, informally, the *rich-gets-richer effect*: if a node has a high degree, then new nodes will likely connect to it, increasing its degree even more. Similarly, low-degree nodes will likely remain with few connections. This very wide distribution leads to the seminal power law degree distribution of scale-free networks:

$$P(D) \propto D^{-\mu} \tag{2.18}$$

where μ represents the decay of the power law: a small μ leads to very large hubs with degree significantly higher than the average degree. Another interesting aspect of scale-free networks is the inherent hierarchical organisation: major hubs are linked to smaller hubs which, in turn, are linked to even minor nodes. A random failure will not lead the network to lose its connectedness, even if an hub is involved (generally, there are more than one hub) [78]. Finally, it is worth mentioning the impact of this hierarchical structure on the clustering coefficient[10]: low-degree nodes tend to form very tight and compact clusters and different clusters are connected via hub nodes. This results in a clustering coefficient which also follows the power law distribution.

This last observation about the clustering coefficient facilitates the introduction of small-world networks. The fact that few hubs can connect small communities implies that the shortest path between any two nodes is rather small (i.e., one node can reach any other node with a relatively limited number of hops). It has been observed that in many networks describing real-world systems the average shortest path length [289] grows very slowly with respect to the number of nodes. This is the so-called *small-world effect*: a network 'seems small' because two nodes are linked by a sequence of very few edges even if they are very far apart. Specifically, the distance H (i.e., the number of hops) between any two randomly chosen nodes grows logarithmically with respect to the number of nodes in the network n:

$$H \propto \log n \tag{2.19}$$

Conversely to the average shortest path length, which is small, the clustering coefficient is high[11]. Small-world networks tend to have highly-connected cliques (in virtue of high clustering coefficient), most pair of nodes connected by short paths (in virtue of small average shortest path length) and there is a consistent number of hubs in order to keep the average number of hops small. The small-world effect is

[10] A measure of the degree to which nodes in a graph tend to cluster together [289].

[11] This is a major striking difference with respect to random graphs: random graphs present small average clustering coefficient and small average shortest path length [376].

even older than Barabási's studies on the World Wide Web: Stanley Milgram in the late 60's conducted the small-world experiment [274], which leaded to the concept of "six degrees of separation". The rules of the experiment were very simple:

1. 296 letters have been sent from random people living in either Omaha (Nebraska, USA) or Wichita (Kansas, USA) to target people living in Boston (Massachusetts, USA)

2. people from Omaha/Wichita were asked whether they knew the target person: if so, they were asked to send the letter directly to the target person; otherwise, the sender had to think about a person that was more likely to know the target and forward the letter to this 'intermediate' person

3. each sender (be it the first or an intermediate one) must write on the letter its name, in order to keep track of the number of steps.

64 letters reached the target person in Boston, with an average path length of 5.5-6 hops, hence the "six degrees of separation". A 'modern' small-world experiment revolves around the so-called "Erdős number", namely the collaborative distance in terms of number of co-authorships between Hungarian mathematician Paul Erdős and another researcher[12].

Figure 2.2 shows an at-the-glance comparison amongst a random graph generated using the Erdős-Rényi model [128], a scale-free network generated using the Barabási-Albert model [25] and a small-world network generated using the Watts-Strogatz model [376].

2.2 Topological Data Analysis

Topological Data Analysis (TDA) is a novel approach suitable whether data can be analysed under a topological point of view [64, 374]. This regards two distinct types of applications [63]:

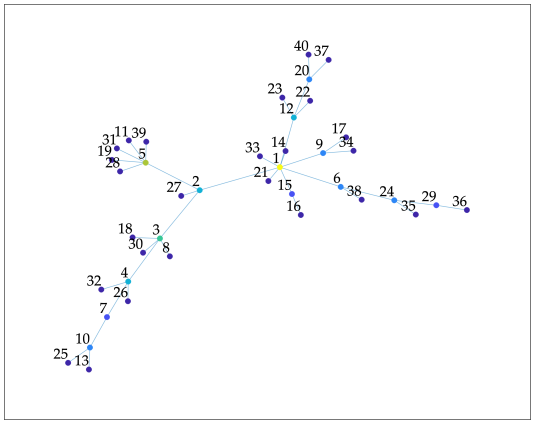
a) when the entire dataset available can be topologically described (e.g., each pattern is a node)

b) when individual entities from the dataset can be topologically described (e.g., each pattern is a graph).

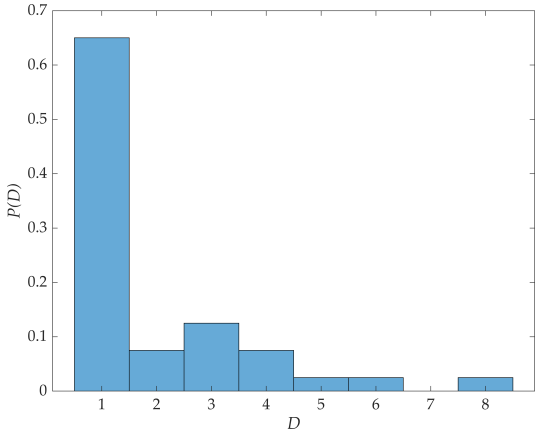
TDA-based techniques regard the use of tools from graph theory, algebraic topology, computational topology and mathematics in order to study the 'shape' of objects lying in topological spaces and in order to extract information from data starting from their topology. Techniques like dimensionality reduction, manifold estimation and persistent homology are commonly used in order to study how components lying in a multi-dimensional space are connected, for example, in terms of loops and multi-dimensional surfaces.

TDA can be applied starting from the so-called *point clouds*, namely a set of points in a multi-dimensional space endowed with a notion of distance or by explicitly providing a similarity matrix between objects. For the sake of generalisation, let us consider the former case and let $\mathcal{C}$ be a given point cloud lying in a given topological

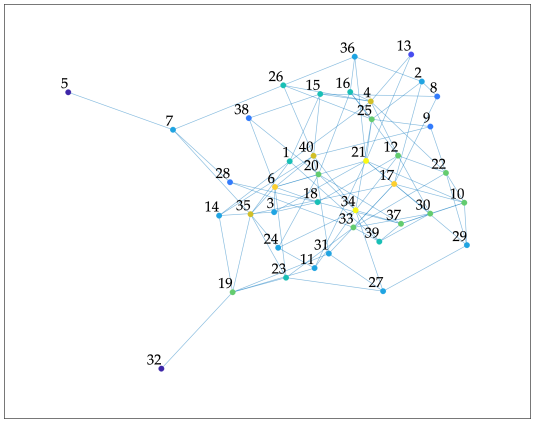
[12] Using the results stored in MathSciNet (https://mathscinet.ams.org/mathscinet/collaborationDistance.html), I was able to estimate my Erdős number as 5 (full path: Alessio Martino → Antonello Rizzi → Witold Pedrycz → Peter J. M. van Laarhoven → Jacobus Hendricus van Lint → Paul Erdős).



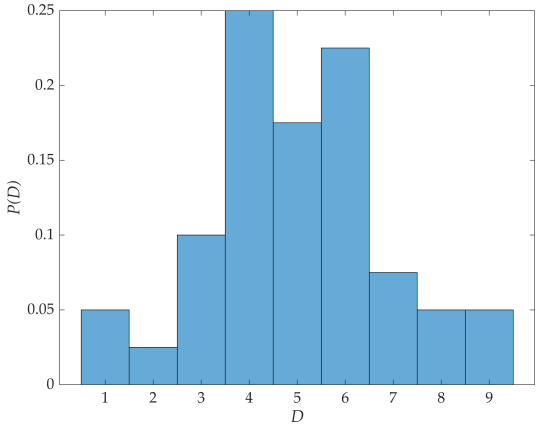
(A) Scale-free (graph)



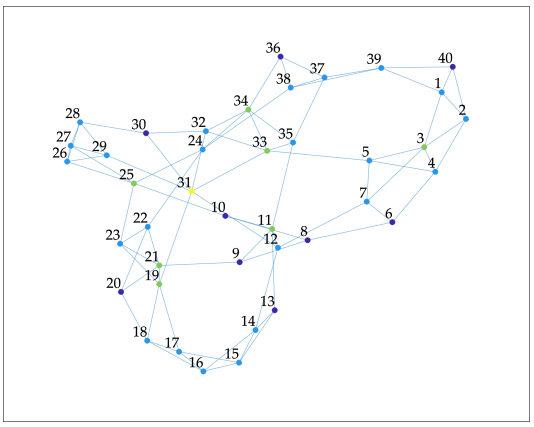
(B) Scale-free (degree distribution)



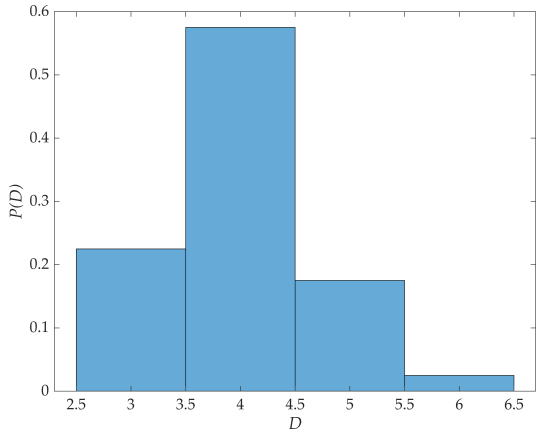
(C) Random (graph)



(D) Random (degree distribution)



(E) Small-world (graph)



(F) Small-world (degree distribution)

FIGURE 2.2: Comparison amongst Random, Scale-free and Small-world graphs. Nodes (left-most plots) are coloured according to their degree from yellow (high degree) to blue (low degree).

space $\mathcal{X}$. Intuitively, in order to study how points in $\mathcal{C}$ are connected, one can study their connectivity in $\mathcal{X}$.

To this end, it is worth defining *simplices* as multi-dimensional topological objects belonging to $\mathcal{X}$, such as points, lines, triangles, tetrahedrons and higher-order analogues, which can be interpreted as topological descriptors of $\mathcal{X}$ itself. Formally

speaking, a k-order simplex (also, k-dimensional simplex or k-simplex) is defined as the convex hull formed by its $k+1$ points (vertices): aforementioned points, lines and triangles are, for example, 0-dimensional, 1-dimensional and 2-dimensional simplices, respectively. Every non-empty subset of the $k+1$ vertices of a k-simplex is a *face* of the simplex: a face is itself a simplex. Trivially, starting from a k-simplex, its faces can be returned via combinatorial extraction. For example, a 2-simplex (triangle) contains three 0-faces (vertices), three 1-faces (lines) and one 2-faces (the triangle itself). Similarly, a 3-simplex (tetrahedron) contains four 0-faces (vertices), six 1-faces (lines), four 2-faces (triangles) and one 3-faces (the tetrahedron itself). More in general, the number of m-faces contained in a k-simplex is given by $\binom{k+1}{m+1}$, hence the total number of faces is given by[13] $2^{k+1}-1$.

Simplicial complexes are properly-constructed finite collection of simplices able to capture the multi-scale organisation (or multi-way relations) in complex systems [27, 28, 169], something that a 'plain graph' cannot represent. Simplicial complexes are an example of *hypergraphs*: a generalisation of 'plain graphs' where *hyperedges* can connect two or more nodes. The modelling limitation offered by graphs can be summarised by the following example [133]: let us represent a scientific collaboration network as a 'plain graph' where nodes corresponds to authors and edges exist whether authors co-authored a paper together. Hence, a group of n co-authors are connected by $n(n-1)/2$ edges with possible ambiguities about whether each pair of authors co-authored a paper or all of the n authors co-authored a paper. Using hypergraphs, an hyperedge can connect the n authors, with no ambiguities. A more biology-oriented example regards protein-protein interaction networks, where nodes corresponds to proteins and edges connect two proteins whether they interact. Similarly, the graph-based representation does not consider protein complexes [146, 246, 317].

Given the above discussion, it is possible to derive the following formal definition.

Definition 1 (Simplicial complex)**.** A simplicial complex is a finite collection of simplices which is closed with respect to the inclusion of the faces and meets the following conditions:

- if a simplex σ belongs to a simplicial complex $\mathcal{S}$, then all faces of σ also belong to $\mathcal{S}$
- for any two simplices $\sigma_1, \sigma_2 \in \mathcal{S}$, their non-empty intersection is a face of both σ_1 and σ_2.

A *subcomplex* of $\mathcal{S}$ is a subset of $\mathcal{S}$ which is also a simplicial complex. An important subcomplex is the k-skeleton, namely a simplicial complex with at most simplices of order k. From this last definition stems another link between 'plain graphs' and simplicial complexes: a 'plain graph' is an example of simplicial complex (1-skeleton) as it contains only points and lines (0-simplices and 1-simplices).

In literature, four main simplicial complexes have been thoroughly investigated [22, 23, 64, 148, 151, 169, 374, 411]:

Alpha Complex: for each point $p \in \mathcal{C}$, evaluate its Voronoi region $V(p)$ (that is, the set of points closest to p). The set of Voronoi regions forms the well-known Voronoi diagram and the nerve of the Voronoi diagram is usually known as

[13] Some authors suggest [410, 411] that the (-1)-simplex, namely the 'empty simplex', is a face of every simplex, augmenting the total number of faces to 2^{k+1}.

Delaunay Complex. For each point $p \in \mathcal{C}$, consider a ball of radius ϵ centred in p and consider the intersection between such ϵ-ball and $V(p)$: this intersection leads to a 'restricted' Voronoi region. The nerve of the restricted Voronoi regions for all points in $\mathcal{C}$ forms the Alpha Complex.

Čech Complex: for each subset of points $P \subset \mathcal{C}$, consider a ball of radius ϵ centred at each point in P; if the balls have non-empty intersection, then P is a valid simplex to be included in the Čech Complex. Dually, the Čech Complex can be defined as the nerve of the balls of radius ϵ centred at each point in $\mathcal{C}$.

Vietoris-Rips Complex: for each subset of points $P \subset \mathcal{C}$, evaluate their pairwise distances: if all pairwise distances are below a given threshold ϵ, then P is a valid simplex to be included in the Vietoris-Rips Complex.

Clique Complex: given an underlying 1-skeleton, the Clique Complex is the simplicial complex built on the top of its maximal cliques. In other words, a k-vertex clique is represented by a $(k-1)$-simplex.

Simplicial homology is a powerful approach in order to study topological spaces characterised by their k-simplices by extending the widely-known number of connected components in a graph to the number of multi-dimensional holes in a simplicial complex. Since simplicial homology only depends on the topological space under analysis, it can be seen as a computable way to distinguish between different topological spaces [10]. The core of simplicial homology are the so-called *Betti numbers*, topological invariants which enumerate the number of holes in each dimension.

Let $\partial_k : \mathcal{S}^{(k)} \rightarrow \mathcal{S}^{(k-1)}$ be the *boundary operator*, namely an incidence-like matrix which maps $\mathcal{S}^{(k)}$ (the set of k-simplices in $\mathcal{S}$) against $\mathcal{S}^{(k-1)}$ (the set of $(k-1)$-simplices in $\mathcal{S}$). The k-order *homology group* is defined as [281]

$$\mathcal{H}_k = \ker\{\partial_k\} / \text{im}\{\partial_{k+1}\} \tag{2.20}$$

where $\ker\{\cdot\}$ and $\text{im}\{\cdot\}$ denote the *kernel* and *image* operators, respectively. The rank of $\mathcal{H}_k$, namely the k^{th} Betti number β_k, is then defined as [169]

$$\beta_k = \text{rank}\{\ker\{\partial_k\}\} - \text{rank}\{\text{im}\{\partial_{k+1}\}\} \tag{2.21}$$

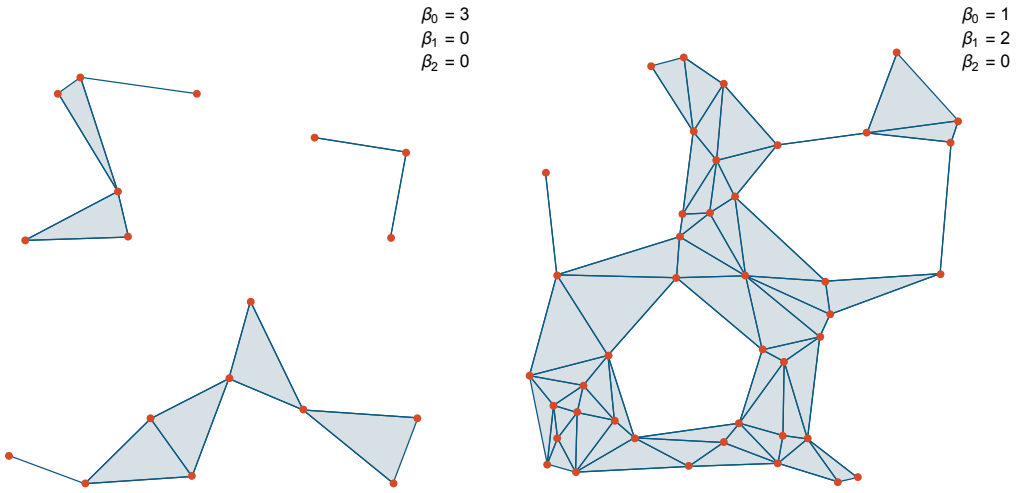
or, thanks to the Rank-Nullity theorem [12]

$$\beta_k = (\text{card}\{\partial_k\} - \text{rank}\{\text{im}\{\partial_k\}\}) - \text{rank}\{\text{im}\{\partial_{k+1}\}\} \tag{2.22}$$

where $\text{card}\{\cdot\}$ corresponds to the cardinality operator (namely, the number of k-simplices) and the rank of the image corresponds to the plain matrix rank in linear algebra. The Betti numbers have the following characteristics:

1. the Betti numbers are all finite
2. the Betti numbers vanish after the spatial dimension
3. the 0^{th} Betti number corresponds to the number of connected components (0-dimensional holes)
4. the 1^{st} Betti number corresponds to the number of circular holes (1-dimensional holes)
5. the 2^{nd} Betti number corresponds to the number of voids/cavities (2-dimensional holes).

Figure 2.3 shows an example of two simplicial complexes, along with their homology.



(A) $\mathcal{S}_1$ (B) $\mathcal{S}_2$

FIGURE 2.3: Example of Simplicial Complexes and their Homology. $\mathcal{S}_1$ (left) has three connected components and no higher-dimensional holes, so the Betti numbers sequence is $(3, 0, 0)$. $\mathcal{S}_2$ (right) has one connected component and two circular holes, so the Betti numbers sequence is $(1, 2, 0)$. Since both simplicial complexes are planar, for both of them $\beta_2 = 0$ is trivial (i.e, there cannot be voids or cavities). Drawn using Wolfram Mathematica 12[14].

[14] http://demonstrations.wolfram.com/SimplicialHomologyOfTheAlphaComplex/

Chapter 3

Pattern Recognition in Structured Domains

3.1 Preliminary Definitions

In Section 1.2, machine learning has been introduced as a powerful tool for complex systems modelling as one very rarely knows a properly-said 'complex system' in closed form. As part of the machine learning umbrella, pattern recognition lies. *Pattern recognition* techniques focus on the classification of objects in a given number of categories (classes). Specifically, pattern recognition includes a wide range of techniques employed to solve (properly said) classification problems and clustering problems. Broadly speaking, pattern recognition techniques can generally be divided in two main families: *supervised* and *unsupervised learning*, both of which (by extension) fall under the data-driven modelling from Section 1.2.

For a more formal definition, let us consider an orientated process $\mathcal{P} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space (domain) and $\mathcal{Y}$ is the output space (codomain). Moreover, let $\{x, y\}$ be a generic input-output sample drawn from $\mathcal{P}$, i.e. $y = \mathcal{P}(x)$.

In supervised learning a finite set $\tilde{\mathcal{P}}$ of input-output pairs drawn from $\mathcal{P}$ are supposed to be known. Common supervised learning tasks can be divided into two families: *classification* and *function approximation* [258]. The two mainly differ on the nature of the output space $\mathcal{Y}$:

- in classification, output values come from a categorical set of possible problem-related classes (e.g., 'healthy' and 'sick' in a predictive medicine problem)
- in function approximation (such as regression, interpolation, fitting, extrapolation), output values usually are drawn from the real numbers field (e.g., predict tomorrow's temperature in a weather forecast problem).

More formally, in the former case, the output space is a non-normed space due to the impossibility to establish any total ordering between its elements or, similarly, due to the nonsensical task of establishing a distance between its elements; whereas, in the latter case, $\mathcal{Y}$ can be considered as a normed space.

In unsupervised learning there are no output values and regularities have to be discovered by considering mutual relations amongst elements belonging to $\mathcal{X}$. One of the mostly acclaimed unsupervised learning approaches relies on *data clustering* [179]. The aim of a clustering algorithm is to discover groups (clusters) of patterns in such a way that similar patterns will fall into the same cluster, whereas dissimilar patterns will fall into different clusters [255–257, 259]. Formally, let $\tilde{\mathcal{P}}$ be a sampling of a non-orientated process $\mathcal{P}$ and let $c \in [2, |\tilde{\mathcal{P}}|]$ be the number of clusters to be

found[15]: a clustering algorithm shall assign, to each $x \in \mathcal{X}$, an integer $h \in [1, c]$ identifying one of c clusters induced over $\tilde{\mathcal{P}}$.

For the sake of completeness, it is worth stressing that clustering and classification algorithms might as well co-operate and shall not be considered as two diametrically opposed techniques. For classification purposes, a rather common approach relies on clustering labelled data without considering their respective labels and then assigning a label to each cluster by considering (for example) the most frequent label amongst the patterns belonging to the cluster itself. Each new pattern can be classified (for example) by inheriting the label of the nearest cluster. An example of such workflow can be found in [111–113].

3.2 Designing a Machine Learning System

In conventional machine learning, a *pattern* is defined by a set of measures related to the original object to be represented, arranged in an array. Each entry (*feature* or *attribute*) is usually a real-valued variable. Geometrically speaking, the pattern can be referred to as *feature vector* and the space spanned by feature vectors is also known as *feature space*. A well-defined feature space is able to facilitate the modelling process: for example, in a classification problem, a well-designed feature space yields simpler decision surfaces in terms of structural complexity (smooth and regular).

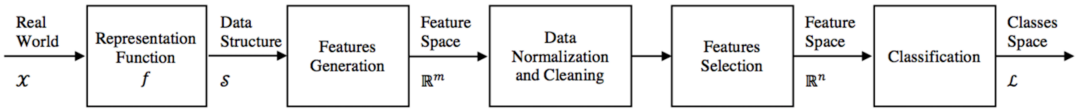


FIGURE 3.1: A (simplified) pattern recognition system workflow.

Let us consider a plain supervised pattern recognition (classification) problem as an instance of the more general data-driven modelling paradigm. Recalling Section 3.1, the aim of a classification problem is to assign to an input pattern, represented by its feature vector, one of the problem-related classes. Figure 3.1 sketches a simplified workflow.

First, real-world data, belonging to a generic and possibly abstract space $\mathcal{X}$ are cast into a proper data structure $\mathcal{S}$, processable by a computational device, by means of a representation function $f(\cdot)$, ad-hoc tailored for the problem at hand. From structured data in $\mathcal{S}$, a given number m of (usually numerical) features is extracted, hence data are cast from $\mathcal{S}$ towards $\mathbb{R}^m$ (i.e., the feature space).

The two following blocks are not mandatory, yet they have been added for the sake of completeness and in order to take into account inevitable uncertainties in data collection and processing. The first block is in charge of data normalisation and cleaning: the former task is sometimes crucial in order to facilitate the (classification) algorithm under particular circumstances[16]; the latter deals with missing and/or noisy data (e.g., outliers' removal[17]). Conversely, the feature selection block

[15] Trivially, it is pointless to find 1 cluster (i.e., the entire dataset) as it is pointless to find more cluster than the number of patterns (i.e., the most degenerate case is where each point is its own cluster.)

[16] Especially when dealing with the Euclidean distance, a common problem is that features spanning a wider range of candidate values have more influence in the resulting distance value, therefore normalising all attributes in the same range (usually $[0, 1]$ or $[-1, +1]$) ensures fair contribution from all attributes, regardless of their original range.

[17] In statistics, *outliers* are 'anomalous data' that, for a given dissimilarity measure, lie far away from most observations.

allows to select a suitable subset of the previously-generated features: indeed, as a general rule, the feature vector should be small, yet informative[18], in order to avoid undesired phenomena such as overfitting and/or the so-called 'curse of dimensionality'. Further, it is recommended to get rid of unreliable features and correlations with existing features.

At the end of this selection stage, feature vectors will lie in a (possibly) reduced space $\mathbb{R}^n$, with $n \leq m$. Finally, feature vectors are used in order to train the classification system, with the final goal of estimating the correct label, an instance of the nominal value set $\mathcal{L} \equiv \mathcal{Y}$.

For a better understanding of Figure 3.1 and all of its steps, let us consider a realistic biology-related scenario about protein classification: $\mathcal{X}$ corresponds the proteins space (i.e., the set of proper macromolecules). Let us suppose to represent proteins as graphs by means of their respective PCNs (see Section 1.3), then $f(\cdot)$ is an hypothetical function which 'converts' macromolecules into graphs. Fortunately, at least from a machine learning viewpoint, molecular biology helps: x-ray crystallography[19], despite it traces back to 1958 [193], is nowadays still the most extensively used technique[20] for experimental evaluation of protein structures[21], making rather easy the evaluation of the corresponding PCNs. The feature generation block aims at extracting numerical features from graphs in $\mathcal{S}$ which, after possible further processing, will directly fed to the classification system.

The training phase for a classification system is a rather delicate task and it needs a separate discussion. Indeed, thanks to the training phase, the classification system learns how to map and discriminate input patterns according to their class labels. In other words, it learns the decision surface (decision region boundaries) which separates patterns corresponding to different classes. A usual procedure for measuring in a fair way the generalisation capability of a classification model consists in splitting the available dataset into two non-overlapping subsets (training set and test set). Specifically, as far as classification problems are concerned, one shall figure both training set and test set as composed by $\{x, y\}$ pairs (see Section 3.1): the classification system, driven by a training algorithm which strictly depends on the chosen model (e.g., Support Vector Machines (SVMs) [50, 86], MultiLayer Perceptron (MLP) [263, 327], K-Nearest Neighbours (K-NN) [87]), uses the training set in order to learn the input-output mapping. The test set will be used on the trained model, with no further adaptive changes, in order to evaluate its performances on previously-unseen patterns (e.g., percentage of patterns correctly classified). This

[18] *Non sunt multiplicanda entia sine necessitate (Entities are not to be multiplied without necessity)*, commonly known as "The Ockham's Razor" Criterion (William of Ockham, circa 1287-1347). This criterion states that, amongst a set of predicting models sharing the same performances, the simplest one (i.e., the one with the simplest decision surface) should be preferred. It is for sure one of the fundamental axioms for thoughtful and practical data-driven modelling.

[19] In short: a beam of incident x-rays is directed towards the crystalline structure under analysis and, by measuring the diffracted beams, one can built a 3D picture of the electron density within the crystal.

[20] Other common techniques include protein nuclear magnetic resonance spectroscopy [385, 386] –in which the 3D structure (namely, how atoms are linked together) is determined by exploiting the mechanical properties of the nucleus of each atom–, mass spectrometry [346] and electron microscopy [282] –in which high-resolution images can be obtained by using electrons as the source of illuminating radiation.

[21] As a matter of fact, recently deposited structures (as of February 2019) in the Protein Data Bank (PDB) database [38] are still experimentally determined via x-ray crystallography. Furthermore, according to PDB, the vast majority of the overall deposited structures have been determined via x-ray crystallography (approx. 125000), the second most common technique being nuclear magnetic resonance (approx. 11000).

double-split procedure *tout court*, however, is not effective since every training algorithm depends on a set of parameters (usually known as *hyper-parameters* in the machine learning terminology) which must be properly tuned in order to maximise the generalisation capabilities of the synthesised model. In order to find a suitable set of hyper-parameters (the so-called *model selection* phase) a three-split procedure is usually employed: the entire dataset is split into three non-overlapping parts (training set, validation set and test set). The training algorithm, driven by a set of hyper-parameters Γ, exploits the training set and its performances are evaluated on the validation set. The parameters Γ are tuned in order to maximise the performances on the validation set and once the (sub-)optimal set $\Gamma^\star$ is found, the final performances are evaluated on the test set. In literature, several ways to perform the aforementioned parameters search have been proposed: amongst which, grid search, random search [37] and (evolutionary) optimisation metaheuristics (e.g., genetic algorithms [153], particle swarm optimisation [194], ant colony optimisation [82] and simulated annealing [199]) emerge. However, if the dataset size is not sufficiently large, a three-split procedure might not be a good choice: indeed, there might not be 'enough patterns' for either the training phase (the model will not learn properly) or the validation/test phase (performances not statistically sound). In this case, the validation set can be derived from the training set via out-of-sample testing, e.g. k-fold cross-validation, leave-one-out, leave-p-out, holdout. By taking k-fold cross-validation [201] as example, the training set is split into k non-overlapping splits (folds) of equal size. Of the k splits, a single split is retained as the validation data for testing the model, and the remaining $k-1$ splits are used as training data. The cross-validation process is then repeated k times, with each of the k folds used exactly once as the validation data. The k results are then averaged to produce a single estimation.

When dealing with unsupervised learning, the conceptual scheme from Figure 3.1 does not change significantly: a Clustering block should be placed in lieu of the Classification block. Recall that a clustering algorithm is in charge of returning a set of clusters according to a given dissimilarity measure and to a predefined objective function. In literature, three main families of clustering algorithms can be found, which mainly differ for their objective function (namely, according to which criterion clusters should be discovered): partitional clustering (e.g., k-means [235, 242], k-medians [51], k-medoids [191, 192]), which split the dataset into k non-overlapping clusters; hierarchical clustering (e.g., BIRCH [402], CURE [160] or several linkage methods [240, 353, 371, 403, 404, 407]), where clusters are found by building a dendrogram in either top-down or bottom-up approach; density-based clustering (e.g., DBSCAN [130, 389], OPTICS [8]), which detect clusters as the most dense regions of the dataset. Clustering algorithms do need hyper-parameters tuning as well. However, since there are no output (ground-truth) values, selecting a (sub)-optimal $\Gamma^\star$ must rely on the so-called internal validation measures [255, 362] such as the Davies-Bouldin index [98] or the Silhouette index [331]. Both manual and fully automatic tuning by means of evolutionary metaheuristics can be employed in unsupervised learning as well.

3.3 Mainstream Approaches

So far, the design of pattern recognition systems has been described in its standard and most common form, that is, where patterns are described by plain real-valued feature vectors. In these cases, any Minkowski-based distance (e.g., Euclidean) can

be a good and straightforward candidate in order to measure (dis)similarity between patterns. Now, it is worth asking the following question:

What if the input space $\mathcal{X} \neq \mathbb{R}^n$?

Before diving into a proper answer, it is worth recalling the following definitions.

Definition 2 (Metric dissimilarity measure)**.** A dissimilarity measure d defined on a generic input space $\mathcal{X}$ is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying the following properties

$$\exists d_0 \in \mathbb{R} \text{ such that } -\infty < d_0 \leq d(x,y) < \infty \tag{3.1}$$
$$d(x,x) = d_0 \tag{3.2}$$
$$d(x,y) = d(y,x) \tag{3.3}$$

for any two objects $x, y \in \mathcal{X}$. If, alongside Eqs. (3.1)–(3.3), d also satisfies the following two properties

$$d(x,y) = d_0 \text{ if and only if } x = y \tag{3.4}$$
$$d(x,z) \leq d(x,y) + d(y,z) \tag{3.5}$$

for any three objects $x, y, z \in \mathcal{X}$, then d is a metric dissimilarity measure.

Definition 3 (Metric similarity measure)**.** A similarity measure s defined on a generic input space $\mathcal{X}$ is a function $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying the following properties

$$\exists s_0 \in \mathbb{R} \text{ such that } -\infty < s(x,y) \leq s_0 < \infty \tag{3.6}$$
$$s(x,x) = s_0 \tag{3.7}$$
$$s(x,y) = s(y,x) \tag{3.8}$$

for any two objects $x, y \in \mathcal{X}$. If, alongside Eqs. (3.6)–(3.8), s also satisfies the following two properties

$$s(x,y) = s_0 \text{ if and only if } x = y \tag{3.9}$$
$$s(x,y) \cdot s(y,z) \leq (s(x,y) + s(y,z)) \cdot s(x,z) \tag{3.10}$$

for any three objects $x, y, z \in \mathcal{X}$, then s is a metric similarity measure.

Under particular circumstances, one can easily 'switch' between (metric) similarity and dissimilarity measures in a given input space, as shown by Theorems 1 and 2. Indeed, dissimilarity measures quantify the degree of separation, whereas similarity measures estimate the complementary notion of closeness.

Theorem 1. *For a given metric dissimilarity measure d, let d_{max} be the maximum pairwise distance between elements in $\mathcal{X}$. Then, $s = d_{max} - d$ is a metric similarity measure.*

Proof. See [356] and [72]. □

Theorem 2. *If d is a metric dissimilarity measure with $d(x,y) > 0$ for any two items $x, y \in \mathcal{X}$, then $s = a/d$ is a metric similarity measure for $a > 0$.*

Proof. See [356]. □

Given the above definitions, it is possible to say that $\mathcal{X}$ is a non-metric (also, non-geometric) space if the pairwise dissimilarities between its elements do not satisfy at

least one of the properties in Eqs. (3.1)–(3.5). This inevitably results in a lack of geometric interpretation of the data, as they cannot be straightforwardly represented in a Euclidean space [230, 232, 309]. Graphs are the seminal example of structured data lying in a non-metric space, especially when nodes and edges themselves contain suitable labels: in this case, graphs convoy not only topological information, but also semantic information about entities and their relations [228, 229]. In literature, five mainstream approaches can be found in order to solve pattern recognition problems in structured domains. Those methods can roughly be divided in three main families, which will be briefly reviewed in the following, highlighting their respective strengths and drawbacks: feature generation / feature engineering (Section 3.3.1), custom dissimilarities in the input space (Section 3.3.2), embedding techniques and kernel methods (Section 3.3.3).

3.3.1 Feature Generation / Feature Engineering

The feature generation strategy quite resembles the simplistic description of Figure 3.1. In this strategy, one defines a mapping function

$$m(\cdot) : \mathcal{X} \to \mathbb{R}^n \tag{3.11}$$

which extracts numerical features from structured data in $\mathcal{X}$ and the n-dimensional feature vector can be fed to any pattern recognition system.

Despite its straightforwardness, this approach deserves a thorough study of the problem at hand: indeed, several numerical characteristics can be extracted from structured data and finding a suitable subset deserves an in-depth investigation (and knowledge) on both the data and the problem to be solved.

3.3.2 Custom Dissimilarities in the Input Space

The feature generation strategy aims at moving the problem from the structured domain $\mathcal{X}$ towards $\mathbb{R}^n$. In the latter, standard pattern recognition techniques can be used without alterations, as it can be equipped with algebraic structures such as the Euclidean distance or the dot product. However, by properly defining a dissimilarity measure on $\mathcal{X}$, this casting procedure can be avoided.

For example, if $\mathcal{X}$ is the input space spanned by strings, the most straightforward example is the Levenshtein distance [76, 221], a generalisation of the well-known Hamming distance [164] which scores the distance between two strings of equal length as the number of mismatches. The Levenshtein distance belongs to the wider family of *edit distances*, in which the distance between any two objects is given by the minimum path of atomic operations in order to transform the two objects into one another. Specifically, the Levenshtein distance is defined as the minimum set of insertions, substitutions and deletions of characters in order to transform the two strings into one another.

Edit distances do not exclusively work in the strings domain: indeed, *graph edit distances* have also been proposed in literature [57–59, 228, 229, 286]. Graph edit distances follow the same rationale behind the plain Levenshtein distance, by defining atomic operations (insertions, substitutions, deletions) on both nodes and edges, with the major drawback of having a non-negligible computational complexity. Indeed, by quoting Horst Bunke [56]:

> *[...] consider the computation of the distance of a pair of objects, which is linear in the number of data items in case of feature vectors, quadratic in case of strings, and exponential for graphs.*

Due to the impossibility to find the optimal sequence of operations with minimum cost, graph edit distances collapse into (possibly greedy) heuristics, e.g. the (node) best-match-first [17, 41, 43, 44]. As per the latter, given two graphs to be matched, say $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$ and $\mathcal{G}_b = (\mathcal{V}_b, \mathcal{E}_b)$, the two sets of nodes are considered first: the first node from $\mathcal{V}_a$ is selected and matched with the most similar[22] node from $\mathcal{V}_b$. The pair is included in the set of node matches, say $\mathcal{M}$, and the two nodes are removed from their corresponding sets. The procedure iterates until $\mathcal{V}_a$ or $\mathcal{V}_b$ is empty. In terms of edit operations, each match counts as a node substitution, whereas insertions and deletions can be deduced by the difference between $|\mathcal{V}_a|$ and $|\mathcal{V}_b|$. The matching procedure then moves towards edges: for each pair of nodes in $\mathcal{M}$, the procedure checks whether an edge between the two nodes exists in $\mathcal{E}_a$ and $\mathcal{E}_b$: if so, this counts as an edge substitution. Otherwise, if the edge exists on $\mathcal{E}_a$, this counts as an edge insertion and, finally, if the edge exists on $\mathcal{E}_b$ only, this counts as an edge deletion.

Amongst these examples, the Levenshtein distance and the Hamming distance are known to be metric, whereas the same might not be true for graph edit distances, which are sensitive to the order of nodes[23] in $\mathcal{V}_a$ and $\mathcal{V}_b$, and they might violate the symmetry property. Furthermore, edit distances are not recommended if the two patterns to be compared have rather different sizes as deletion and insertion operations can easily prevail over substitutions.

On the plus side, methods based on ad-hoc (dis)similarities notably work in cases where the pattern recognition system does not need to define an algebraic structure on the input space. This, however, has the intrinsic drawback of limiting the number of algorithms (both supervised and unsupervised) which can be employed. For example, let us consider a clustering task to be performed into a non-metric space with a given (dis)similarity measure: algorithms such as k-means or k-medians cannot be considered as prospective candidates since, in order to update the cluster prototype, the former needs to evaluate the component-wise mean amongst the patterns belonging to the cluster itself, whereas the latter needs to evaluate the component-wise median. In both cases, the need to define an algebraic structure emerges which, however, turns into a non-sense as non-metric spaces are concerned. Suitable clustering algorithms for such scenarios are k-medoids, as discussed in [255–257], and the Basic Sequential Algorithmic Scheme (BSAS) [356], whether equipped with the MinSoD (medoid) representative [107], namely the element of the cluster which minimises the sum of pairwise distances. Similarly, as classification problem are concerned, a straightforward candidate is the K-NN as it classifies test patterns according to the pairwise distances with respect to some already-known data. Indeed, the distance measure can easily be tailored to the data at hand, see e.g. [118], and might as well not rely on operators such as the dot product, mandatory in MLPs or linear SVMs.

[22] According to a dissimilarity measure defined on the node labels.

[23] In order to make the matching procedure more robust, usually a given number of matching procedures are performed by shuffling nodes in $\mathcal{V}_a$ and $\mathcal{V}_b$ and the match which leads to the minimum distance determines the overall distance between $\mathcal{G}_a$ and $\mathcal{G}_b$.

3.3.3 Embedding Techniques

Embedding strategies are popular techniques in order build a feature space in which pattern recognition algorithms can be applied without alternations. The major difference between embedding techniques and the feature generation approach described in Section 3.3.1 is that the latter case considers a one-to-one mapping between patterns in the original space and their respective feature vectors (each pattern is individually transformed thanks to the mapping function $m(\cdot)$). Conversely, in the former case, the feature space (embedding space) is built by considering some sort of 'cooperation' and exploiting relations between patterns, or substructures drawn from them.

Dissimilarity Space

The dissimilarity space embedding can be summarised by this simple, yet effective, idea: each pattern is represented according to the pairwise distances with respect to all other patterns [125, 309].

Specifically, let $\mathcal{D}$ be a dataset whose patterns belong to a structured domain $\mathcal{X}$ and let $d(\cdot,\cdot)$ be a dissimilarity function defined over $\mathcal{X}$. The embedding towards a dissimilarity space consists in defining the following dissimilarity matrix:

$$\mathbf{P}_{i,j} = d(x_i, x_j) \quad \forall x_i, x_j \in \mathcal{D} \tag{3.12}$$

and then using the i^{th} row (or column) from $\mathbf{P}$ in lieu of the i^{th} pattern from $\mathcal{D}$. In other words, each pattern is described as a $|\mathcal{D}|$-dimensional real-valued vector containing the pairwise distances with respect to all other patterns, including itself.

Two caveats have to be considered. First, if $d(\cdot,\cdot)$ does not satisfy the symmetry property, then $\mathbf{P}$ will not be symmetric either and ambiguities exist about whether to represent the i^{th} pattern according to the i^{th} row or column of $\mathbf{P}$. In this case, it is possible to 'enforce' symmetry by letting

$$\mathbf{P} \leftarrow \frac{1}{2}\left(\mathbf{P} + \mathbf{P}^T\right) \tag{3.13}$$

where $(\cdot)^T$ denotes the transpose operator.

Second, building the dissimilarity space as in Eq. (3.12) has time and space complexity which goes as $\mathcal{O}(c \cdot |\mathcal{D}|^2)$, where c indicates the computational complexity for evaluating a single dissimilarity between any two patterns, and can be troublesome for large datasets and/or computationally expensive dissimilarity measures $d(\cdot,\cdot)$. In order to overcome this problem, in [310], it has been proposed to choose a suitable subset of prototypes $\mathcal{R} \subset \mathcal{D}$ as pivotal patterns in order to build the embedding space. Therefore, it is possible to define a 'reduced' dissimilarity space as follows:

$$\overline{\mathbf{P}}_{i,j} = d(x_i, x_j) \quad \forall x_i \in \mathcal{D}, x_j \in \mathcal{R} \tag{3.14}$$

Evaluating the pairwise distances with respect to the prototypes drops the overall complexity from $\mathcal{O}(c \cdot |\mathcal{D}|^2)$ to $\mathcal{O}(c \cdot |\mathcal{D}| \cdot |\mathcal{R}|)$. The price to pay for lower complexity relies on a thoughtful and suitable choice of patterns to be included in $\mathcal{R}$: the cardinality of $\mathcal{R}$ impacts the complexity, yet the selected representatives must well characterise the decision boundary. For carrying out the delicate task of electing patterns in $\mathcal{R}$, several techniques have been proposed in literature, including [100, 231]:

- random selection, possibly class-aware in case of supervised problems

- clustering procedures over $\mathcal{D}$ using $d(\cdot,\cdot)$ (cf. Section 3.3.2) in order to use the resulting medoids (MinSoDs) as prototypes
- optimisation-driven techniques, where a suitable subset of prototypes is selected in order to optimise a given objective function (e.g., classifier performances on some validation/test data).

Information Granulation

Another explicit embedding procedure relies on Granular Computing (GrC). GrC is a human-inspired information processing paradigm suitable for modelling complex systems that explores multiple levels of 'granularity' in data [29, 305, 394]. The concept of GrC arose from many branches of natural and social sciences [172, 393], and it is at the basis of recently developed frameworks in computational intelligence [225, 345]. GrC aims at the extraction of suitable entities known as *information granules* which are strictly problem- and data-dependent. The process of 'granulation', intended as the extraction of meaningful data aggregates, mimics the human mechanism needed to organise complex data from the surrounding environment in order to support decision making activities and describe the world around [306].

The importance of information granules resides in the ability to underline properties and relationships between data aggregates. Specifically, their synthesis can be achieved by following the *indistinguishability rule*, according to which elements that show enough similarity, proximity or functionality shall be grouped together [398]. With this approach, each granule is able to show homogeneous semantic information from the problem at hand [307]. Furthermore, data at hand can be represented using different levels of granularity and thus different peculiarities of the system can emerge. When analysing a system with high level of detail, one shall expect a huge number of very compact information granules since finer details are of interest. Conversely, as the level of abstraction increases, the number of information granules decreases and, as a result, one shall expect very few, yet populated and less compact information granules. Depending on the resolution, a problem may exhibit different properties and different atomic units that show different representations of the whole system.

Due to the very tight link between "information granules" and "groups-of-similar-data", data clustering is one of the mainstream approaches in order to extract information granules from a set of (possibly structured) data [119]. In fact, data clustering has been successfully employed in several GrC-based classification systems for structured data, including graphs [17, 41, 43, 44, 231] and sequences [243, 315, 323, 324].

All aforementioned works share four macro-blocks for synthesising the pattern recognition system. The first block is the "extractor" which returns sub-structures[24] drawn from the training data. The resulting set of sub-structures is forwarded to the next module, the "granulator": a clustering algorithm (usually BSAS or BSAS-like) equipped with an ad-hoc (dis)similarity suitable to the domain (e.g., graph edit distance in case of graphs or string matching techniques [183, 221] in case of strings/documents) returns a suitable number of clusters. The corresponding MinSoDs form an *alphabet* of pivotal symbols on the top of which the embedding can be performed thanks to the "embedder" block: let $\mathcal{A} = \{s_1, \ldots, s_M\}$ be the alphabet of M symbols, each (structured) pattern x is cast towards an M-length integer-valued

[24] e.g., simple paths in case of graphs or adjacent items (characters, words) in case of sequences.

vector $\mathbf{h} \in \mathbb{R}^M$ defined as

$$\mathbf{h}_{\mathcal{A}}(x) = [\text{count}(s_1, x), \ldots, \text{count}(s_M, x)] \tag{3.15}$$

where $\text{count}(a, b)$ is a function that enumerates a in b. $\mathbf{h}$ is the so-called *symbolic histogram* [105, 106], as it contains the number of occurrences of each alphabet symbol within the original pattern. The space spanned by symbolic histograms is the embedding space, which can be equipped with any algebraic structure, and the "classifier" block, usually driven by K-NN equipped with the plain Euclidean distance, takes care of performing the classification. Those advanced pattern recognition systems for structured domains are usually driven by genetic algorithms in order to tune weights and parameters involved, including the cluster radius and maximum number of allowed clusters for BSAS, weights for the (dis)similarity measure (if any) and the binary mask for feature selection.

The feature selection phase deserves a few additional remarks due to the following benefits:

1. the clustering procedure can easily return symbols which might not be useful for the classification task: discarding unpromising symbols can improve the classification performances and avoid phenomena such as the curse of dimensionality or overfitting

2. the complexity (dimensionality) of the embedding space is reduced and the classification of new patterns is faster (i.e., less symbols to match with – see Eq. (3.15))

3. GrC-based pattern recognition systems allow an interesting a-posteriori knowledge discovery phase: depending on the application, one can let field-experts to analyse the information granules that somehow 'survived' the feature generation phase in order to gather further insights on the modelled system.

Kernel Methods

Dissimilarity spaces and GrC-based techniques rely on explicit embeddings towards a vector space: in the former case, the embedding space is built on top of (a possibly reduced subset of) the training data; in the latter case, the embedding is built by considering meaningful and recurrent sub-structures extracted from the training data. Further, in both cases, an a-posteriori knowledge discovery phase is possible: in the former case, one can analyse the patterns included in the prototypes set; in the latter case, one can analyse the resulting alphabet.

Kernel methods are another powerful pattern analysis technique that, conversely, relies on an implicit embedding and can be used with suitable learning systems, the seminal example being SVMs [86]. Kernel methods rely on the so-called *kernel functions* [271] to implicitly embed the input data towards an high-dimensional (possibly infinite-dimensional) Hilbert space $\mathcal{H}$. In the latter, according to the Cover's Theorem [88], linear classification is more likely.

Definition 4 ((Positive Definite) Kernel)**.** Let $\mathcal{X}$ be a generic input space and let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous function from the product space $\mathcal{X} \times \mathcal{X}$ towards the reals. The function $K(\cdot, \cdot)$ is called a (positive definite) kernel if it is symmetric

and positive definite, that is:

$$K(x_i, x_j) = K(x_j, x_i) \qquad \forall x_i, x_j \in \mathcal{X} \tag{3.16}$$

$$\sum_{i=1}^{N_P} \sum_{j=1}^{N_P} c_i c_j K(x_i, x_j) \geq 0 \quad \forall c_i, c_j \in \mathbb{R}, \forall x_i, x_j \in \mathcal{X} \tag{3.17}$$

Usually kernel methods are employed whether the input space has an underlying Euclidean geometry: indeed, the simplest kernel (the *linear kernel*) is the plain dot product between feature vectors

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{3.18}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product.

Let us consider the input data to be arranged in an $N \times m$ instance matrix, where N denotes the number of patterns and m denotes the number of features. The kernel matrix[25] $\mathbf{K} \in \mathbb{R}^{N \times N}$ can also be evaluated in a batch fashion as

$$\mathbf{K} = \mathbf{X} \cdot \mathbf{X}^T \tag{3.19}$$

If data points are linearly separable, the linear kernel can lead to satisfactory results. Otherwise, one can replace the dot product with one of the many kernel functions available. In Table 3.1 some popular non-linear kernels suitable for dealing with vector data are summarised.

TABLE 3.1: Some popular non-linear kernels for vector data.

Kernel	Definition	Parameters
Radial Basis Function	$K(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma \|\mathbf{x} - \mathbf{y}\|^2\}$	
Polynomial	$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + v)^p$	$v \geq 0$, $p \geq 0$
Hyperbolic Tangent	$K(\mathbf{x}, \mathbf{y}) = \tanh\{a\mathbf{x} \cdot \mathbf{y} + r\}$	$a > 0$, $r < 0$
Laplace	$K(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}\right\}$	

The power of kernel methods can be summarised by the *kernel trick* [335], described be the following, seminal equation:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{3.20}$$

which basically states that instead of performing an explicit mapping thanks to a (possibly unknown and possibly computationally expensive) function $\phi(\cdot)$ and then performing the dot product between the two transformed vectors, one can define a kernel function $K(\cdot, \cdot)$ satisfying the Mercer's condition, namely Eqs. (3.16)–(3.17), that takes care of implicitly performing both embedding and dot product in $\mathcal{H}$.

So far, kernel methods have been described for dealing with vector data, stressing the fact that kernels should satisfy the Mercer's condition. However, especially in the 2000s, a lot of research has been devoted towards learning from empirical (indefinite) kernels, namely 'kernels' that do not satisfy the Mercer's condition. As SVMs are concerned, a non-positive definite kernel does not guarantee the maximal-margin optimisation problem to be convex, hence local solutions can be returned. Nonetheless, some empirical kernels have been proved to be quite successful, or

[25] Also known as the *Gram matrix*, after Danish mathematician Jørgen Pedersen Gram.

some techniques have been proposed in order to foster an empirical kernel matrix towards positive definiteness, see e.g. [73, 74, 163, 210, 296].

In the same timespan, research has been devoted to the development of kernel functions for dealing with structured data as well. Notable examples include strings [219, 220], images [207] and, obviously, graphs. As the latter case is concerned, graph kernels can be divided into four big families [147]:

- *Model driven kernels* exploit generative models or transformative models. An example of generative model driven kernel is the Fisher kernel [175], which exploits hidden Markov models. Conversely, an example of transformative model driven kernel is the diffusion kernel [203], which exploits the heat equation in order to diffuse local information (neighbourhood) to all the graph.

- The largest family is composed by *syntax driven kernels*, which aim at analysing trees, paths, cycles or subgraphs. Notable examples include random walk kernels [145, 366], shortest path kernels [49], graphlet kernels [342] and the Weisfeiler–Lehman kernels [341, 343]. The first three, respectively, take into consideration the number of random walks between two graphs via their product graph, the number of common shortest paths and the number of common k-order graphlets. The fourth one, as instead, computes the number of subtree-walks shared between two graphs using the Weisfeiler–Lehman test of graph isomorphism.

- Another state-of-the-art approach is given by *propagation kernels* [288], which are based on the spread of information across several graphs using early-stage distributions from propagation schemes (e.g., random walks) in order to model different types of node and edge labels. Such propagation schemes make propagation kernels suitable for dealing also with unlabelled and partially labelled graphs.

- Finally, *deep graph kernels* [391] can be seen as an extension of the aforementioned syntax driven kernels, where a positive semidefinite matrix weights (encodes) the relationships between sub-structures (e.g., graphlets).

Chapter 4

Proposed Pattern Recognition Systems

4.1 Graph Classification by Spectral Density Estimation

In Section 2.1, the main graph matrix representations have been introduced, namely the adjacency matrix $\mathbf{A}$, the degree matrix $\mathbf{D}$, the Laplacian matrix $\mathbf{L}$ and the normalised Laplacian $\overline{\mathbf{L}}$. Again from Section 2.1, it is worth recalling the following characteristics of the normalised Laplacian matrix:

1. conveys many structural and dynamical properties of the network
2. is a square and symmetric matrix: as such, there exist a spectral decomposition with non-complex eigenvectors and eigenvalues
3. following point 2, there are no negative eigenvalues and, specifically, all eigenvalues lie in range $[0, 2]$.

This first experiment concentrates on the normalised Laplacian matrix, with the latter point serving as the main actor.

Let $\mathcal{D} = \{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$ be a dataset of N graphs of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For the sake of generalisation, let us suppose that graphs in $\mathcal{D}$ have different sizes. For each graph, one can easily evaluate the four matrices $\mathbf{A}, \mathbf{D}, \mathbf{L}$ and $\overline{\mathbf{L}}$; however, if the graph has $n = |\mathcal{V}|$ nodes, then $\mathbf{A}, \mathbf{D}, \mathbf{L}, \overline{\mathbf{L}} \in \mathbb{R}^{n \times n}$, hence none of these matrices can directly be used in order to compare two graphs having different number of nodes.

Let us consider the spectral decomposition of $\overline{\mathbf{L}}$ as in Eq. (2.8):

$$\overline{\mathbf{L}} = \mathbf{V} \mathbf{\Lambda}^{(\overline{L})} \mathbf{V}^T \tag{4.1}$$

with $\mathbf{\Lambda}^{(\overline{L})} = \text{diag}\left\{\lambda_1^{(\overline{L})}, \ldots, \lambda_n^{(\overline{L})}\right\}$ being a diagonal matrix containing the eigenvalues of $\overline{\mathbf{L}}$. Despite the aforementioned boundedness in $[0, 2]$, the size of the spectrum (i.e., the number of eigenvalues) equals the number of nodes: so neither the spectrum can directly be used in order to compare two graphs having different sizes.

Following [244], it is possible to estimate the graph spectral density by means of a kernel density estimator [303] with Gaussian kernel. In other words, the (normalised) Laplacian matrix is treated as a random matrix with a well-defined spectral density $p(x)$ [158, 180, 181, 267, 269] that reads as:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x - \lambda_i)^2}{2\sigma^2}} \tag{4.2}$$

where σ denotes the bandwidth. In order to consider a bandwidth that automatically scales in a graph-wise fashion, the Scott's rule [338] has been considered, defined as

$$\sigma = \frac{3.5\hat{\sigma}}{n^{-1/3}} \tag{4.3}$$

where $\hat{\sigma}$ denotes the sample standard deviation and n denotes its size.

Let $\mathcal{G}_a$ and $\mathcal{G}_b$ two graphs to be compared and let $p_a(x)$ and $p_b(x)$ be their respective spectral densities. Their distance can be evaluated by taking the ℓ_2 norm:

$$d(\mathcal{G}_a, \mathcal{G}_b) = \int_0^2 (p_a(x) - p_b(x))^2 dx \tag{4.4}$$

The same operation can be performed in the discrete domain by sampling a finite number of samples m, equal for all graphs, from their respective spectral densities: each graph is therefore unambiguously identified by a vector in $\mathbb{R}^m$ and the distance between any graphs, formerly Eq. (4.4), collapses into a plain Euclidean distance.

Figure 4.1 summarises the steps required in order to evaluate the spectral density of a given graph. Starting from an unweighted and undirected graph $\mathcal{G}$ with $n = 20$ nodes (Figure 4.1a) described by its adjacency matrix $\mathbf{A}$ (Figure 4.1b), the degree matrix $\mathbf{D}$, the Laplacian matrix $\mathbf{L}$ and the normalised Laplacian matrix $\overline{\mathbf{L}}$ are evaluated (Figures 4.1c, 4.1d and 4.1e, respectively) thanks to Eqs. (2.2), (2.4) and (2.7), respectively. Figure 4.1f closes the pre-processing chain: the eigenvalues of $\overline{\mathbf{L}}$ are computed first (red crosses on the x-axis), in order to evaluate the spectral density as in Eq. (4.2) (smooth blue curve) and, eventually, a given number of $m = 100$ samples linearly spaced in $[0, 2]$ are drawn from the resulting distribution (blue stems).

By recalling the three-fold partition of techniques for solving pattern recognition problems in structured domains (Section 3.3), it is safe to say that this approach falls under the feature generation umbrella (Section 3.3.1). The herein described pre-processing chain has to be individually performed on each graph in $\mathcal{D}$ in order to cast each graph into an m-length feature vector. By following the standard machine learning convention, the dataset $\mathcal{D}$ can be transformed into an instance matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$ with patterns (i.e., feature vectors) arranged as rows. This instance matrix, lying in a Euclidean space, can be freely fed to any machine learning system.

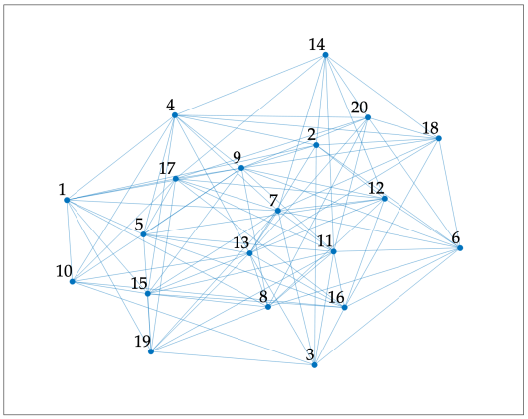
4.2 Graph Classification using the Betti Numbers Sequence

In Section 2.2, TDA has been introduced along with the definition of simplices, simplicial complexes, simplicial homology and Betti numbers. Specifically, Betti numbers have been defined as topological invariants in order to distinguish different topological spaces.

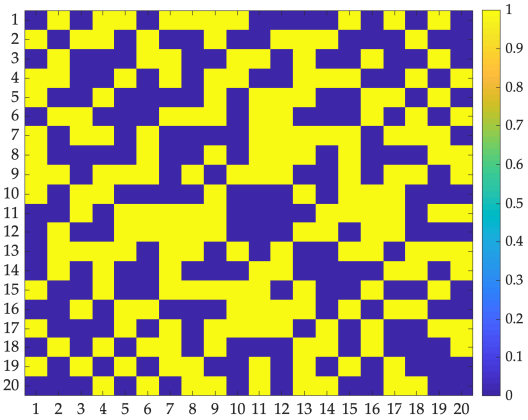
The evaluation of the Betti numbers starts from the definition of a properly constructed simplicial complex. In this case, the Vietoris-Rips complex is considered due to lighter computational burden and intuitiveness. In [410] a fast three-steps procedure for building the Vietoris-Rips is proposed:

1. given a set of points $\mathcal{C}$ equipped with a notion of distance $d(\cdot, \cdot)$, the Vietoris-Rips neighbourhood graph $\mathcal{G}_{\text{VR}} = (\mathcal{V}, \mathcal{E})$ is build by considering $\mathcal{C} \equiv \mathcal{V}$ and by adding edges in $\mathcal{E}$ whether two points are less than or equal to ϵ apart, according to $d(\cdot, \cdot)$

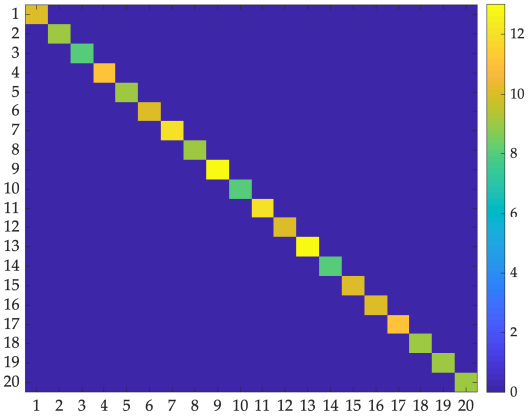
2. the Clique Complex of $\mathcal{G}_{\text{VR}}$ is evaluated (i.e., thanks to the Bron-Kerbosch algorithm [54])



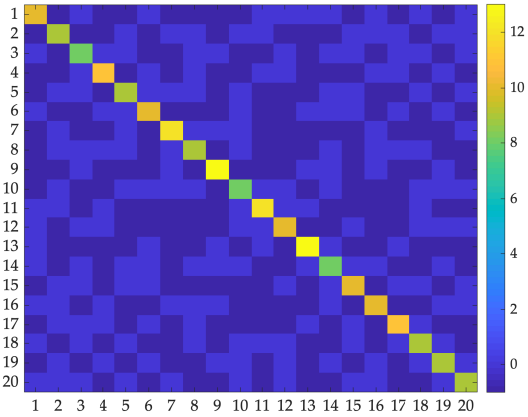
(A) A (random) graph



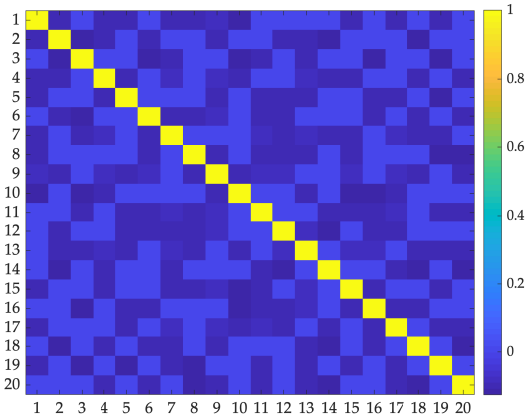
(B) Adjacency Matrix



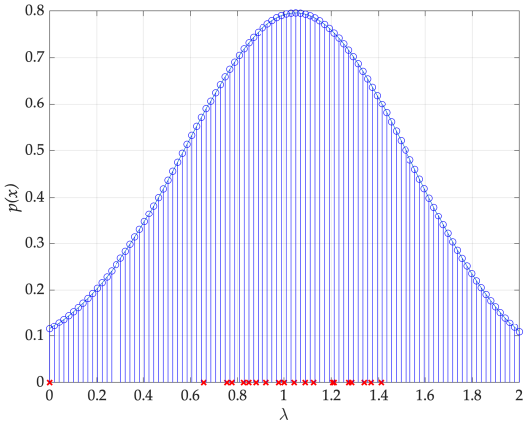
(C) Degree Matrix



(D) Laplacian Matrix



(E) Normalised Laplacian Matrix



(F) (Sampled) Spectral Density

FIGURE 4.1: Pre-processing chain for evaluating graphs spectral density.

3. the faces of the (maximal) simplices (i.e., lower order simplices) can be extracted combinatorially.

Given a suitable value for ϵ, once the Vietoris-Rips is built, the Betti numbers can be evaluated thanks to Eqs. (2.20)–(2.22).

The choice of ϵ is critical as it determines the 'resolution' of the simplicial complex. In Figure 4.2 an example is shown which sees the 1HNR protein (already in Figure 1.1) at different scales. Usually, in TDA one finds a suitable value for ϵ thanks

to persistence diagrams and barcodes [71, 412], useful visualisation tools for studying relevant topological features. The idea behind persistence can be summarised as follows: as ϵ changes, new connected components can appear, disappear, can be merged to other connected components. In higher dimensions, voids and cavities can appear, can be filled, and so on. Persistence is a tool which 'tracks' the lifetime of such changes and, with the rationale that features having longer lifetimes are more important, one finds a suitable scale value that lets such important features to be 'visible'. The data analysis task proceeds using this suitable value. Figure 4.3 shows the barcode for protein 1HNR (Figure 4.2): the barcode shows the persistence of topological features as ϵ changes. There is a strong link between the barcode and the Betti numbers: the number of k-dimensional holes, namely β_k, at a given scale $\overline{\epsilon}$ is the number of intervals in dimension k that intersects a vertical line through $\overline{\epsilon}$. For example, see Figure 4.3, at $k = 0$ there are plenty of bars crossing $\epsilon \in [0, 4)$, meaning that there are plenty of 0-dimensional holes, hence the graph is strongly disconnected with a lot of connected components (coherently with the PCN connectivity range – cf. Section 1.3). For $\epsilon \geq 4$ there is only one interval at $k = 0$, meaning that there is one connected component. Further analogies between barcodes and Betti numbers can be found in the caption of Figure 4.3.

The learning approach herein discussed considers the rationale behind persistence by putting it in a feature engineering scenario. Specifically, let $\mathcal{D} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ be a dataset of N point clouds lying in an m-dimensional space and let $\epsilon_1, \ldots, \epsilon_j$ be a set of j suitable candidates for the scale parameter ϵ.

For each point cloud in $\mathcal{D}$ the Vietoris-Rips complex is evaluated by considering each of the ϵ candidate values and, consequently, its Betti numbers are computed. Hence, a given point cloud $\mathcal{C}$ is cast towards an integer-valued vector defined as:

$$\left[\beta_0^{(\epsilon_1)}, \ldots, \beta_{m-1}^{(\epsilon_1)}, \ldots, \beta_0^{(\epsilon_j)}, \ldots, \beta_{m-1}^{(\epsilon_j)}\right] \tag{4.5}$$

where $\beta_a^{(b)}$ is the a^{th} Betti number at $\epsilon = b$. In other words, the feature vector collects the number of 'holes' in each dimension as ϵ changes.

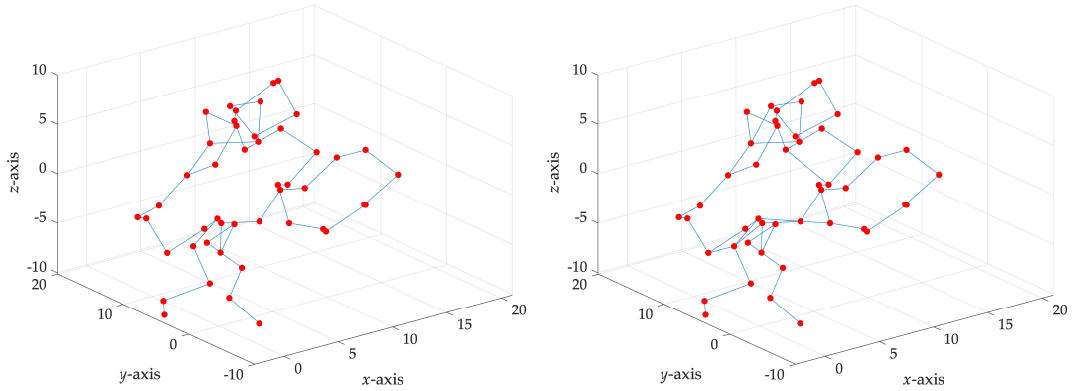
The dataset $\mathcal{D}$ can therefore be transformed into an instance matrix $\mathbf{X} \in \mathbb{R}^{N \times (j \cdot m)}$ which spans a multidimensional Euclidean space that can trivially be equipped with any algebraic structure for pattern recognition purposes.

4.3 Embedding over Simplicial Complexes

In Section 2.2, simplicial complexes have been defined as finite collections of simplices that, in turn, have been introduced as 'multidimensional building blocks' of a topological space. Further, in Section 4.2, the Betti numbers have been feature-engineered as prospective features for pattern recognition purposes. The technique herein described leverages on the following question:

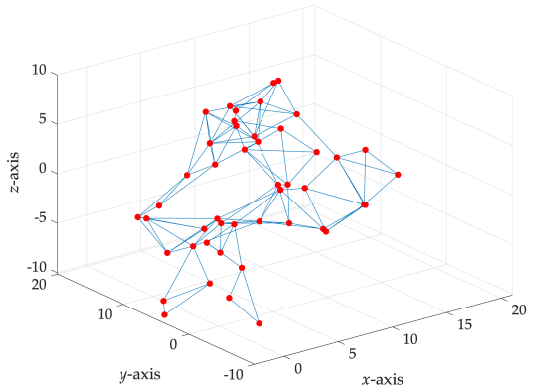
Can simplices be interpreted as meaningful information granules?

Specifically, recalling the GrC-based embedding techniques from Section 3.3.3, the possibility of using these topological objects as pivotal substructures in order to build an embedding space for pattern recognition purposes is investigated. Furthermore, this experiment not only marks a significant deviation from previous two feature engineering strategies (Sections 4.1 and 4.2) towards an embedding via information granulation, but also sees semantic information playing a crucial role. Indeed, in Section 2.1, graphs have been described as abstract entities able to capture

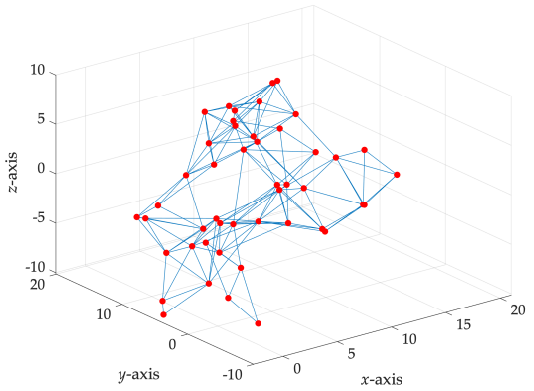


(A) $\epsilon = 4$

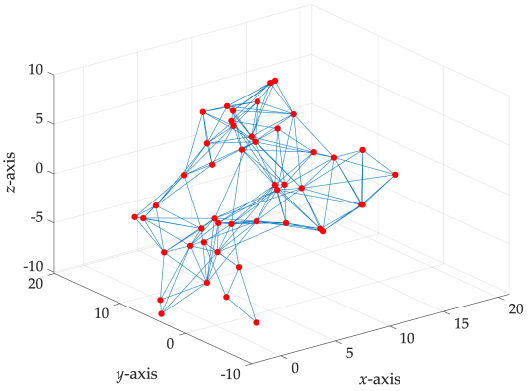
(B) $\epsilon = 5$



(C) $\epsilon = 6$



(D) $\epsilon = 7$



(E) $\epsilon = 8$

FIGURE 4.2: Topological configuration of a sample PCN (PDB 1HNR) at different scales. For the sake of completeness, at $\epsilon = 4$ the sequence of Betti numbers is $\beta_0 = 1$, $\beta_1 = 1$, $\beta_2 = 0$; at $\epsilon = 5$ one gets $(1, 7, 0)$; at $\epsilon = 6$ one gets $(1, 4, 0)$; at $\epsilon = 7$ one gets $(1, 3, 0)$ and, finally, at $\epsilon = 8$ one gets $(1, 2, 0)$.

both semantic and topological information: nonetheless, in the two previous strategies, labels on nodes and edges have not been taken into account.

Let $\mathcal{D}$ be a dataset of graphs[26] (1-skeletons) of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}_v)$, where $\mathcal{L}_v$

[26] In Chapter 5 this embedding procedure will be tested using the Clique Complex in order to exploit the topological information already available in the considered datasets. However, it is worth noting that any simplicial complex can be used instead of the Clique Complex, so this embedding technique can easily be extended towards datasets composed by point clouds.

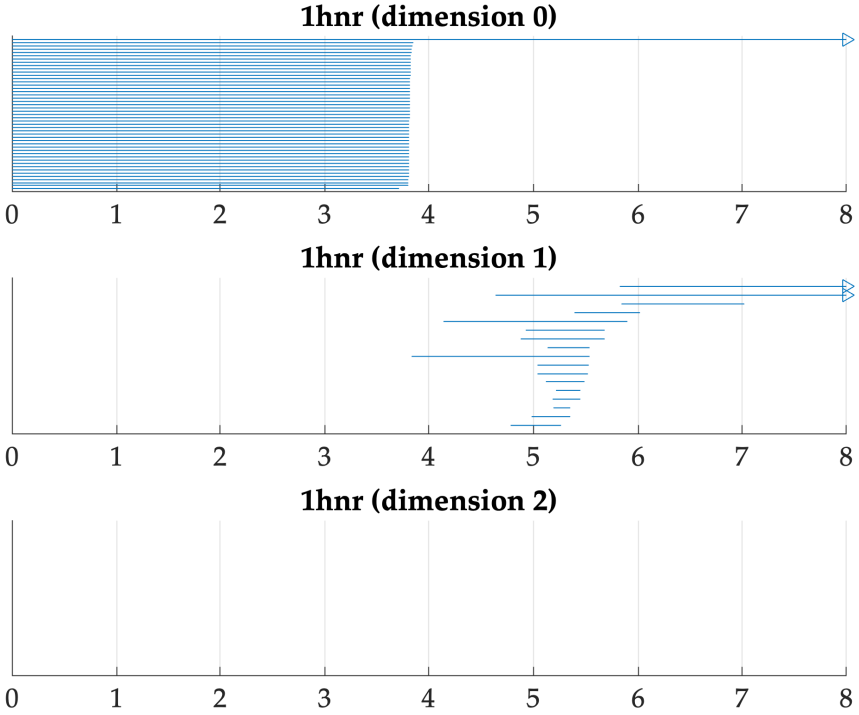


FIGURE 4.3: Barcode for sample PCN (PDB 1HNR) drawn using JavaPlex [354] for MATLAB®. The filtration value ϵ lies on the x-axis. For example, note that, at $\epsilon = 4$, one 0-dimensional interval and one 1-dimensional interval cross the corresponding vertical line, while there are no intervals in dimension 2: the Betti numbers are therefore $(1,1,0)$. Similarly, at $\epsilon = 5$, one 0-dimensional interval and seven 1-dimensional intervals cross the corresponding vertical line, while there are no intervals in dimension 2: the Betti numbers are $(1,7,0)$. These 'plain counts' perfectly match the Betti numbers sequences from Figure 4.2, which have been evaluated analytically according to Eqs. (2.20)–(2.22).

is the set of node labels: for the sake of ease, let us consider node labels belonging to a finite categorical set (e.g., strings). Finally, let $\mathcal{D}$ be split into three disjoint subsets, namely training set ($\mathcal{D}_{\text{TR}}$), validation set ($\mathcal{D}_{\text{VAL}}$) and test set ($\mathcal{D}_{\text{TS}}$), i.e. such that:

$$\mathcal{D}_{\text{TR}} \cup \mathcal{D}_{\text{VAL}} \cup \mathcal{D}_{\text{TS}} = \mathcal{D} \tag{4.6}$$

$$\mathcal{D}_{\text{TR}} \cap \mathcal{D}_{\text{VAL}} = \emptyset, \qquad \mathcal{D}_{\text{TS}} \cap \mathcal{D}_{\text{VAL}} = \emptyset, \qquad \mathcal{D}_{\text{TR}} \cap \mathcal{D}_{\text{TS}} = \emptyset \tag{4.7}$$

$$\mathcal{D}_{\text{TR}} \cap \mathcal{D}_{\text{VAL}} \cap \mathcal{D}_{\text{TS}} = \emptyset \tag{4.8}$$

The first step consists in inferring the simplicial complexes separately for all graphs in the three splits:

$$\mathcal{D}_{\text{TR}}^{\text{SC}} = \{sc(\mathcal{G}),\ \forall \mathcal{G} \in \mathcal{D}_{\text{TR}}\} \tag{4.9}$$

$$\mathcal{D}_{\text{VAL}}^{\text{SC}} = \{sc(\mathcal{G}),\ \forall \mathcal{G} \in \mathcal{D}_{\text{VAL}}\} \tag{4.10}$$

$$\mathcal{D}_{\text{TS}}^{\text{SC}} = \{sc(\mathcal{G}),\ \forall \mathcal{G} \in \mathcal{D}_{\text{TS}}\} \tag{4.11}$$

where $sc(\cdot) : \mathcal{G} \rightarrow \mathcal{S}$ is a function that builds the simplicial complex $\mathcal{S}$ starting from the 1-skeleton $\mathcal{G}$.

The embedding considers the simplices belonging to the simplicial complexes in $\mathcal{D}_{\text{TR}}^{\text{SC}} \cup \mathcal{D}_{\text{VAL}}^{\text{SC}}$. In information granulation terms, the union of these simplices forms the alphabet on the top of which the embedding can be performed. This operation, however, has the following two drawbacks:

1. in case of large networks and/or large datasets, this might lead to a huge number of simplices (very high-dimensional space)

2. simplices are impossible to match, hence evaluating the symbolic histograms is impossible. In order to focus this point, let us consider any given vertex belonging a given graph to be identified by a unique progressive number. In this case, it is impossible to match two simplices belonging to possibly different simplicial complexes (i.e., determine whether they are equal or not): one can easily say whether they share the same order by looking at the number of nodes belonging to the simplex, yet one cannot say whether they share the same nodes.

In order to overcome these problems, node labels play a crucial role. Each node can dually be identified by its node label drawn from $\mathcal{L}_v$, other than a progressive unique identifier. This conversion from 'simplices-of-nodes' to 'simplices-of-node-labels' has a three-fold meaning:

1. matching two simplices (possibly belonging to two different simplicial complexes) is straightforward: two simplices are equal if they share the same order and their respective vertices share the same node labels

2. the enumeration of different, unique, simplices is straightforward

3. by means of this conversion, simplicial complexes are *de facto* treated as multi-sets: indeed, within the same simplicial complex, different simplices can share the same node labels.

It is possible to define the three counterparts of Eqs. (4.9)–(4.11) where each given node v belonging to a given simplex σ is identified by its node label:

$$\overline{\mathcal{D}}^{\text{SC}}_{\text{TR}} = \{\mathcal{L}_v(v),\ \forall v \in \sigma,\ \forall \sigma \in \mathcal{S},\ \forall \mathcal{S} \in \mathcal{D}^{\text{SC}}_{\text{TR}}\} \tag{4.12}$$

$$\overline{\mathcal{D}}^{\text{SC}}_{\text{VAL}} = \{\mathcal{L}_v(v),\ \forall v \in \sigma,\ \forall \sigma \in \mathcal{S},\ \forall \mathcal{S} \in \mathcal{D}^{\text{SC}}_{\text{VAL}}\} \tag{4.13}$$

$$\overline{\mathcal{D}}^{\text{SC}}_{\text{TS}} = \{\mathcal{L}_v(v),\ \forall v \in \sigma,\ \forall \sigma \in \mathcal{S},\ \forall \mathcal{S} \in \mathcal{D}^{\text{SC}}_{\text{TS}}\} \tag{4.14}$$

Let $\mathcal{A}$ be the alphabet composed by the union of the distinct (unique) simplices in $\overline{\mathcal{D}}^{\text{SC}}_{\text{TR}} \cup \overline{\mathcal{D}}^{\text{SC}}_{\text{VAL}}$: as per the symbolic histograms technique, $\mathcal{A}$ contains the set of pivotal substructures (i.e., information granules – cf. Section. 3.3.3) on the top of which the embedding is performed. Let $M = |\mathcal{A}|$ be the number of symbols, each simplicial complex $\mathcal{S}$ is transformed into an M-length integer-valued vector (the symbolic histogram – cf. Eq. (3.15)) as follows:

$$\mathbf{h}_{\mathcal{A}}(\mathcal{S}) = [\text{count}(\mathcal{A}_1, \mathcal{S}), \ldots, \text{count}(\mathcal{A}_M, \mathcal{S})] \tag{4.15}$$

Thanks to the symbolic histograms embedding, the three sets $\overline{\mathcal{D}}^{\text{SC}}_{\text{TR}}$, $\overline{\mathcal{D}}^{\text{SC}}_{\text{VAL}}$ and $\overline{\mathcal{D}}^{\text{SC}}_{\text{TS}}$ are transformed into three instance matrices $\mathbf{X}_{\text{TR}} \in \mathbb{R}^{|\mathcal{D}_{\text{TR}}| \times M}$, $\mathbf{X}_{\text{VAL}} \in \mathbb{R}^{|\mathcal{D}_{\text{VAL}}| \times M}$ and $\mathbf{X}_{\text{TS}} \in \mathbb{R}^{|\mathcal{D}_{\text{TS}}| \times M}$ lying in a Euclidean space and suitable to be processed by standard pattern recognition algorithms.

4.4 Embedding via INDVAL

With this embedding technique the focus returns to 'plain graphs' rather than hypergraphs and simplicial complexes. Furthermore, conversely to the previous case

from Section 4.3, this embedding procedure exploits the ground-truth labels from the classification problem at hand.

The core of this procedure is the so-called INDVAL score: a sensitivity/specificity integrated evaluation approach originally proposed in [124] and developed for individuating the 'signature species' of a given environment. The philosophy at the basis of the INDVAL is straightforward: a given species, say S, is 'representative' (thus useful for the recognition of a given environmental condition E) if it satisfies both of the following properties

1. S must be present only (or almost only) in the E-positive objects (e.g., fields, samples of territory)

2. S must be present in all (or the great majority) of the E-positive cases.

Rather than individuating 'signature species' of different environments, in this embedding technique, the INDVAL is used in order to spot 'signature substructures' in structured data. These 'signature substructures' will finally form the alphabet of pivotal substructures for building the embedding space.

As per the previous case, let $\mathcal{D}$ be a dataset of graphs where each graph has the form $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}_v)$ and let $\mathcal{L}$ be the set of ground-truth class labels for graphs in $\mathcal{D}$. Again, for the sake of ease, let us suppose that node labels $\mathcal{L}_v$ belong to a finite and categorical set. Let $\mathcal{D}_{\text{TR}}$, $\mathcal{D}_{\text{VAL}}$ and $\mathcal{D}_{\text{TS}}$ be three disjoint subsets drawn from $\mathcal{D}$ satisfying Eqs. (4.6)–(4.8) and, finally, consider $\mathcal{L}$ to be split accordingly ($\mathcal{L}_{\text{TR}}$, $\mathcal{L}_{\text{VAL}}$ and $\mathcal{L}_{\text{TS}}$).

In [124], the authors proposed a unified index (INDVAL) aimed at identifying the sensitivity and the specificity of single individual elements (species, in their work) belonging to different groups (environments, in their work). As the procedure herein described is concerned, "individual elements" can be substructures of structured data (e.g., edges in a graph), whereas "groups" can be deduced thanks to $\mathcal{L}$. Given the definition of substructure (e.g., edge), under the graphs point of view, the INDVAL I can be restated as follows:

$$A_{i,j} = \frac{\text{\# patterns in class } j \text{ having edge } i}{\text{\# patterns having edge } i} \tag{4.16}$$

$$B_{i,j} = \frac{\text{\# patterns in class } j \text{ having edge } i}{\text{\# patterns belonging to class } j} \tag{4.17}$$

$$I_{i,j} = A_{i,j} \cdot B_{i,j} \cdot 100 \tag{4.18}$$

By definition, since $A_{i,j} \in [0, 1]$ and $B_{i,j} \in [0, 1]$, then $I_{i,j} \in [0, 100]$. The three involved players (A, B, I) have the following interpretation:

- A is a measure of specificity: its maximum value is obtained when the i^{th} edge can be found only in patterns of class j

- B is a measure of sensitivity: its maximum value is obtained if all patterns of class j have the i^{th} edge

- the INDVAL I considers the two previous terms: the maximum INDVAL corresponds to the maximum specificity and sensitivity of the i^{th} edge for the j^{th} group (class), namely the 'perfect edge', which is contained in all patterns of class j and is absent in all patterns belonging to other classes.

Let $\overline{\mathcal{E}}$ be the set of unique edges[27] in $\mathcal{D}_{\text{TR}} \cup \mathcal{D}_{\text{VAL}}$ and let $\overline{\mathcal{L}}$ be the set of unique ground-truth class labels (i.e., the number of classes for the classification problem at hand), then one can figure $\mathbf{A}, \mathbf{B}, \mathbf{I} \in \mathbb{R}^{|\overline{\mathcal{E}}| \times |\overline{\mathcal{L}}|}$ as compact matrix representations of Eqs. (4.16), (4.17) and (4.18), respectively. Then, one can select edges (rows) from $\mathbf{I}$ whether at least one of the INDVALs (columns) is above a user-defined threshold T. In other words, edges are selected if considered 'relevant' for at least one of the problem-related classes. Hence, the alphabet $\mathcal{A} = \{s_1, \ldots, s_M\}$ is composed by the set of selected edges.

As per the previous case, the embedding follows the symbolic histogram approach: each graph is described by an M-length integer-valued vector containing the number of occurrences of each relevant edge within the original graph. The three sets ($\mathcal{D}_{\text{TR}}$, $\mathcal{D}_{\text{VAL}}$ and $\mathcal{D}_{\text{TS}}$) are finally transformed into the three already familiar instance matrices ($\mathbf{X}_{\text{TR}} \in \mathbb{R}^{|\mathcal{D}_{\text{TR}}| \times M}$, $\mathbf{X}_{\text{VAL}} \in \mathbb{R}^{|\mathcal{D}_{\text{VAL}}| \times M}$ and $\mathbf{X}_{\text{TS}} \in \mathbb{R}^{|\mathcal{D}_{\text{TS}}| \times M}$).

4.5 Hypergraph Kernels

In Sections 4.1–4.2 two feature engineering techniques have been proposed (Section 3.3.1) and in Sections 4.3–4.4 two GrC-based embedding strategies have been presented (Section 3.3.3). In this Section (and in the following one), kernel methods will be of interest.

In Section 3.3.3 the main families of graph kernels have been introduced, so it is worth asking the following question:

Is it possible to define kernels over simplicial complexes (hypergraphs)?

The overall setup is the same as the embedding over simplicial complexes described in Section 4.3: let us consider a dataset $\mathcal{D}$ composed by either point clouds or 1-skeletons. Let the vertices be equipped with categorical node labels.

The first step is to evaluate the simplicial complexes for all items in $\mathcal{D}$: trivially, if $\mathcal{D}$ is composed by 1-skeletons, one can directly use the Clique Complex; otherwise, if $\mathcal{D}$ is composed by point clouds, one can use the Alpha Complex or the Vietoris-Rips Complex[28]. As already discussed in Section 4.3, let each vertex belonging to a given simplex to be identified by its corresponding node label in order to easily match two simplices: two simplices can be considered equal if they share the same order and the same node labels. Given the simplicial complexes, the following four hypergraph kernels are proposed [254].

The *Histogram Kernel* (HK) is loosely based on the symbolic histogram technique. Let $\mathcal{S}_i$ and $\mathcal{S}_j$ be two simplicial complexes and let $\mathcal{A} = \mathcal{S}_i \cup \mathcal{S}_j$ be the set of unique simplices belonging to either $\mathcal{S}_i$ or $\mathcal{S}_j$. Then, a given simplicial complex, say $\mathcal{S}$, can be cast into a vector, say $\mathbf{f} \in \mathbb{R}^{|\mathcal{A}|}$, where

$$\mathbf{f}_k = \text{count}(\mathcal{A}_k, \mathcal{S}), \qquad \forall k = 1, \ldots, |\mathcal{A}| \tag{4.19}$$

where $\text{count}(a, b)$ is a function that counts the number of occurrences of a in b. Hence, thanks to Eq. (4.19), $\mathcal{S}_i \to \mathbf{f}^{(i)}$ and $\mathcal{S}_j \to \mathbf{f}^{(j)}$ and HK has the form

$$K_{\text{H}}(\mathcal{S}_i, \mathcal{S}_j) = \langle \mathbf{f}^{(i)}, \mathbf{f}^{(j)} \rangle \tag{4.20}$$

[27] The unique edges can be enumerated by considering the two nodes (or, better, their labels) at their extremities (alike to the simplices case in Section 4.3).

[28] Due to heavy computational burden, using the Čech Complex is always discouraged.

However, HK defined as in Eq. (4.20) can be skewed by the different number of simplices within each simplicial complex. To this end, the following normalisation is adopted:

$$K_{\mathrm{H}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(j)} \rangle}{\sqrt{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(i)} \rangle \cdot \langle \mathbf{f}^{(j)}, \mathbf{f}^{(j)} \rangle}} \tag{4.21}$$

From Section 2.2 it is clear that simplicial complexes as sets, hence a straightforward idea is to use a similarity measure between sets as the core of the kernel. The seminal example is the Jaccard similarity [176–178], defined as the ratio between the size of the intersection divided by the size of the union between two sets. Since a simplex can be identified by the set of labels associated to each node, different simplices within the same simplicial complex can share the same node labels. This means that simplicial complexes are *de facto* multisets and the Jaccard similarity as previously defined loses its effectiveness. For dealing with multisets, the weighted Jaccard similarity (also known as Ružička index [332]) is considered. Given two vectors, say $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, the weighted Jaccard similarity is defined as

$$J_W(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}{\sum_{i=1}^{n} \max(\mathbf{a}_i, \mathbf{b}_i)} \tag{4.22}$$

and the *Weighted Jaccard Kernel* (WJK) is therefore defined as

$$K_{\mathrm{WJ}}(\mathcal{S}_i, \mathcal{S}_j) = J_W(\mathbf{f}^{(i)}, \mathbf{f}^{(j)}) \tag{4.23}$$

where $\mathbf{f}^{(i)}$ and $\mathbf{f}^{(j)}$ follow from Eq. (4.19).

The *Edit Kernel* (EK) aims at measuring the similarity between two simplicial complexes according to the number of simplices to be inserted/removed/substituted in order to transform the two simplicial complexes into one another. Let $e(\mathcal{S}_i, \mathcal{S}_j)$ be an edit distance (Levenshtein-like) with unitary weights. The first step is to convert the distance measure into a (possibly normalised) similarity measure. In [396] it has been demonstrated that

$$\bar{e}(\mathcal{S}_i, \mathcal{S}_j) = \frac{2 \cdot e(\mathcal{S}_i, \mathcal{S}_j)}{|\mathcal{S}_i| + |\mathcal{S}_j| + e(\mathcal{S}_i, \mathcal{S}_j)} \tag{4.24}$$

is a normalised edit distance in range $[0, 1]$ which satisfies the properties of a metric. From Section 3.3, recall that if d is a normalised metric distance, then $1 - d$ is a normalised metric similarity. Hence, EK has the form

$$K_{\mathrm{E}}(\mathcal{S}_i, \mathcal{S}_j) = 1 - \bar{e}(\mathcal{S}_i, \mathcal{S}_j) \tag{4.25}$$

It is noteworthy that edit distances/similarities are sensitive to the order of the input sequences. In order to ease the matching procedure, within each simplicial complex, simplices are jointly sorted both lexicographically (i.e., according to the node labels) and according to their orders.

The *Stratified Edit Kernel* (SEK) takes into account the following issue with EK: the latter can be skewed if the two simplicial complexes have a high variety of simplices per order. Let $\mathcal{K}$ be the set of different orders amongst simplices in the two simplicial complexes to be matched, then the SEK is defined as

$$K_{\mathrm{SE}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} 1 - \bar{e}(\mathcal{S}_i^{(k)}, \mathcal{S}_j^{(k)}) \tag{4.26}$$

where, recall, $\mathcal{S}^{(k)}$ denotes the subset of k-simplices in the simplicial complex $\mathcal{S}$. The stratification allows to treat independently subsets of simplices having the same order; nonetheless, simplices are sorted lexicographically within each subset.

By their respective definitions, the four kernels exploit simplicial complexes under different lights: for HK, an explicit embedding towards a vector space is performed before evaluating the linear kernel; WJK relies on the set (or, better, multiset) structure of the simplicial complexes when equipped with semantic information on their respective nodes; EK and SEK measure the similarity in terms of edit operations defined on simplices.

In Section 3.3.3 the importance of positive definiteness has been stressed, so it is worth addressing whether the four proposed kernels satisfy the Mercer's condition.

Theorem 3. *The Histogram Kernel is positive definite.*

Proof. The Histogram Kernel is based on pairwise dot products, hence it trivially satisfies Mercer's condition, as already in Eq. (3.18). Furthermore, the normalisation adopted does not affect positive definiteness, as stated by Schölkopf and Smola in [335]. □

Theorem 4. *The Weighted Jaccard Kernel is positive definite.*

Proof. In order show that the Weighted Jaccard Kernel is a valid kernel is worth mentioning the following theorem (the proof can be found in [144]):

Let $K(a,b)$ be a kernel function satisfying Mercer's condition, then the following kernel is still valid:

$$\hat{K}(a,b) = \frac{K(a,b)}{K(a,a) + K(b,b) - K(a,b)} \tag{4.27}$$

Eq. (4.22) can be re-written as follows

$$J_W(\mathbf{a},\mathbf{b}) = \frac{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}{\sum_{i=1}^{n} \mathbf{a}_i + \sum_{i=1}^{n} \mathbf{b}_i - \sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)} \tag{4.28}$$

and by considering Eq. (4.27), Eq. (4.28) can be re-written as

$$J_W(\mathbf{a},\mathbf{b}) = \frac{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{a}_i) + \sum_{i=1}^{n} \min(\mathbf{b}_i, \mathbf{b}_i) - \sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)} \tag{4.29}$$

By comparing Eqs. (4.27) and (4.29), it is clear that the Weighted Jaccard Kernel is a valid kernel if $K(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)$ is a valid kernel as well. In order to demonstrate the latter, it is worth mentioning the following lemma (the proof can be found in [283]):

Let $\{a_1, \ldots, a_m\}$ be a finite set of real-valued numbers, then the matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$, defined as follows, is a valid kernel matrix:

$$\mathbf{M}_{i,j} = \min(a_i, a_j) \tag{4.30}$$

As the Weighted Jaccard Kernel deals with real-valued vectors in $\mathbb{R}^n$ and not with real-valued scalars as in the previous lemma, we shall extend the lemma to the former case. To this end, it is possible to build a series of n matrices, where each matrix (of the form as in Eq. (4.30)) considers the pairwise minimum along a given dimension, say $i = 1, \ldots, n$, of the considered vectors and then perform the element-wise sum of such matrices:

$$K(\mathbf{a},\mathbf{b}) = \mathbf{M}^{(1)} + \ldots + \mathbf{M}^{(i)} + \ldots + \mathbf{M}^{(n)} \tag{4.31}$$

Finally, Eq. (4.31) can easily be shown to be a valid kernel matrix thanks to the property that the element-wise sum of positive definite matrices leads to a positive definite matrix, as shown by Horn and Johnson in [171]. □

Theorem 5. *The Edit Kernel and the Stratified Edit Kernel are not positive definite.*

Proof. Edit distances (similarities) are well-known to lead to possibly indefinite kernels, as shown by Neuhaus and Bunke in [287] and stated by Riesen and Bunke in [320]. □

4.6 Graph Classification using a Multiple Kernel Approach

In this last technique, different graph representations are simultanously considered by an hybridisation of multiple kernel learning and dissimilarity representations.

Let us start by briefly recalling the rationale behind dissimilarity spaces (Section 3.3.3). Let $\mathcal{D} = \{x_1, \ldots, x_N\}$ be a dataset lying in a given input space $\mathcal{X}$, not necessarily metric, and let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a dissimilarity measure defined over $\mathcal{X}$. The dataset $\mathcal{D}$ can be cast towards a dissimilarity space by building a dissimilarity matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$ as follows:

$$\mathbf{P}_{i,j} = d(x_i, x_j) \quad \forall x_i, x_j \in \mathcal{D} \tag{4.32}$$

In order to reduce the computational complexity of the dissimilarity space embedding, one can think of selecting a given prototype subset $\mathcal{R} \subset \mathcal{D}$ and perform the embedding towards a 'reduced' dissimilarity space $\bar{\mathbf{P}} \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{R}|}$ by considering the pairwise dissimilarities with respect to the prototypes only [310]:

$$\bar{\mathbf{P}}_{i,j} = d(x_i, x_j) \quad \forall i = 1, \ldots, |\mathcal{D}|, \; \forall j = 1, \ldots, |\mathcal{R}| \tag{4.33}$$

In Section 3.3.3, kernel methods have also been introduced as powerful pattern recognition technique, especially for solving non-linear problems. In *multiple kernel learning* [155] a given number of N_K kernels are properly combined into a final kernel matrix $\mathbf{K}$, with the linear combination being the most straightforward option:

$$\mathbf{K} = \sum_{i=1}^{N_K} \omega_i \mathbf{K}^{(i)} \tag{4.34}$$

where $\mathbf{K}^{(i)}$ are single 'sub-kernels'. The weights ω_i can be learned according to different strategies and can be constrained in several ways, see e.g. [14, 85, 154, 155, 174, 208, 222, 351]. Typically, multiple kernel methods are employed according to two different strategies:

- same data to be analysed by several kernels, which may differ in shape and/or type
- different data (possibly coming from different sources) to be analysed by the same (or different) kernels.

In this case, a mixed approach is considered: same source (graphs), but different representations. Further, a convex linear combination of radial basis function kernels is considered, hence

$$\mathbf{K}^{(i)}_{j,k} = \exp\left\{-\gamma_i \cdot \|x_j - x_k\|^2\right\} \qquad \forall j, k = 1, \ldots, N \tag{4.35}$$

where γ_i is the shape parameter (see Table 3.1), and the kernel weights ω_i are constrained as

$$\sum_{i=1}^{N_K} \omega_i = 1 \tag{4.36}$$

$$\omega_i \in [0,1] \qquad \text{for } i = 1, \dots, N_K \tag{4.37}$$

Before diving into an in-depth description of the proposed technique, is worth stressing the following.

Theorem 6. *A multiple kernel machine as in Eq. (4.34) where each single kernel has the form Eq. (4.35) and where kernels are weighted as in Eqs. (4.36)–(4.37) satisfies Mercer's condition.*

Proof. Recall from Section 3.3.3 that the radial basis function kernel is a valid kernel (i.e., satisfying Mercer's condition). In Cristianini and Shawe-Taylor [90], it has been shown that the summation of two valid kernels is still a valid kernel, whereas in Horn and Johnson [171] it has been shown that a positive semi-definite matrix multiplied by a non-negative scalar returns a positive semi-definite matrix. By merging these results the proof is straightforward, and the proposed multiple kernel can be freely used on kernelised SVMs. □

Let $\mathcal{D}$ be a structured dataset, to be considered as split in three non-overlapping training, validation and test set ($\mathcal{D}_{\text{TR}}$, $\mathcal{D}_{\text{VAL}}$ and $\mathcal{D}_{\text{TS}}$, respectively). Especially in structured domains, several representations (set of descriptors) might hold for the same pattern and let $\{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N_R)}\}$ be such representations, split in the same fashion (i.e. $\{\mathbf{X}^{(i)}_{\text{TR}}\}_{i=1}^{N_R}$, $\{\mathbf{X}^{(i)}_{\text{VAL}}\}_{i=1}^{N_R}$ and $\{\mathbf{X}^{(i)}_{\text{TS}}\}_{i=1}^{N_R}$). Finally, let $\{d_1(\cdot,\cdot), \dots, d_{N_R}(\cdot,\cdot)\}$ be the set of dissimilarity measures, ad-hoc chosen for each of the N_R representations. Hence, one can evaluate the pairwise dissimilarity matrices between training, validation and test set:

$$\begin{aligned}
\mathbf{P}^{(1)}_{\text{TR}} &= d_1(\mathbf{X}^{(1)}_{\text{TR}}, \mathbf{X}^{(1)}_{\text{TR}}) & \dots \quad \mathbf{P}^{(N_R)}_{\text{TR}} &= d_{N_R}(\mathbf{X}^{(N_R)}_{\text{TR}}, \mathbf{X}^{(N_R)}_{\text{TR}}) \\
\mathbf{P}^{(1)}_{\text{VAL}} &= d_1(\mathbf{X}^{(1)}_{\text{VAL}}, \mathbf{X}^{(1)}_{\text{TR}}) & \dots \quad \mathbf{P}^{(N_R)}_{\text{VAL}} &= d_{N_R}(\mathbf{X}^{(N_R)}_{\text{VAL}}, \mathbf{X}^{(N_R)}_{\text{TR}}) \\
\mathbf{P}^{(1)}_{\text{TS}} &= d_1(\mathbf{X}^{(1)}_{\text{TS}}, \mathbf{X}^{(1}_{\text{TR}}) & \dots \quad \mathbf{P}^{(N_R)}_{\text{TS}} &= d_{N_R}(\mathbf{X}^{(N_R)}_{\text{TS}}, \mathbf{X}^{(N_R)}_{\text{TR}})
\end{aligned} \tag{4.38}$$

Let $\mathbf{w} \in \{0,1\}^{|\mathcal{D}_{\text{TR}}|}$ be a binary vector in charge of selecting columns from the dissimilarity matrices in Eq. (4.38). Therefore, $\mathbf{w}$ allows to move from the 'full' dissimilarity space towards the 'reduced' dissimilarity space (Eq. (4.32) vs. Eq. (4.33)) or, in other words, $\mathbf{w}$ acts as prototype selector:

$$\begin{aligned}
\bar{\mathbf{P}}^{(1)}_{\text{TR}} &= \mathbf{P}^{(1)}_{\text{TR}}(:, \mathbf{w}) & \dots \quad \bar{\mathbf{P}}^{(N_R)}_{\text{TR}} &= \mathbf{P}^{(N_R)}_{\text{TR}}(:, \mathbf{w}) \\
\bar{\mathbf{P}}^{(1)}_{\text{VAL}} &= \mathbf{P}^{(1)}_{\text{VAL}}(:, \mathbf{w}) & \dots \quad \bar{\mathbf{P}}^{(N_R)}_{\text{VAL}} &= \mathbf{P}^{(N_R)}_{\text{VAL}}(:, \mathbf{w}) \\
\bar{\mathbf{P}}^{(1)}_{\text{TS}} &= \mathbf{P}^{(1)}_{\text{TS}}(:, \mathbf{w}) & \dots \quad \bar{\mathbf{P}}^{(N_R)}_{\text{TS}} &= \mathbf{P}^{(N_R)}_{\text{TS}}(:, \mathbf{w})
\end{aligned} \tag{4.39}$$

From the 'reduced' dataset it is possible to train a multikernel ν-SVM where the kernel matrix has the form as in Eq. (4.34) and each 'sub-kernel' has the form as in Eq. (4.35), hence

$$\mathbf{K} = \sum_{i=1}^{N_R} \omega_i \cdot \exp\left\{-\gamma_i \cdot \|\bar{\mathbf{P}}^{(i)}_{\text{TR}} \ominus \bar{\mathbf{P}}^{(i)}_{\text{TR}}\|^2\right\} \tag{4.40}$$

where $\ominus$ denotes the pairwise distances.

From Eq. (4.40) it is clear that each 'sub-kernel' operates on a different representation ($N_K \equiv N_R$). Further, albeit there is some research about learning from indefinite kernels (see e.g. [73, 74, 163, 210, 296]), namely kernels that do not satisfy Mercer's condition, the evaluation on the top of Euclidean spaces (e.g., dissimilarity spaces) retain the (conditionally) positive definiteness, devoting matrix regularisation or other tricks to foster positive definiteness.

The combination of training and validation set helps in tuning the several parameters involved, namely the regularisation term for SVMs, the kernel shapes γ_i, the kernel weights ω_i and the feature selector vector $\mathbf{w}$.

4.7 Final Remarks

In this Chapter, six techniques for graph-based pattern recognition have been introduced and discussed.

The first two techniques, respectively described in Sections 4.1 and 4.2, aim at describing a given graph according to its (sampled) spectral density and according to the sequence of Betti numbers at different scales. Both of these techniques fall under the feature generation and feature engineering umbrella (see Section 3.3.1). Amongst the two techniques, the spectral density-based is the easiest to evaluate: indeed, the computational complexity required for generating features out of a single graph $\mathcal{G}$ with n nodes is upper-bounded by $\mathcal{O}(n^2)$, namely the computational complexity of the QR algorithm for returning the QZ factorisation (also known as generalised Schur decomposition) for solving the generalised eigenvalue problem. Conversely, the evaluation of the k^{th} Betti number (for a given scale parameter value) has complexity which goes like $\mathcal{O}(pq^2) + \mathcal{O}(qr^2)$ if p is the number of $(k-1)$-simplices, q is the number of k-simplices and r is the number of $(k+1)$-simplices, namely the complexities of two singular value decompositions for evaluating the two matrix ranks (see Eq. (2.22)). For a given scale value, one must also consider the computational complexity required for building the simplicial complex whose Betti numbers are of interest.

The other two techniques, respectively described in Sections 4.3 and 4.4, aim at describing a given graph via symbolic histograms using the GrC paradigm. In the former case, simplices are considered as candidate information granules, whereas in the latter case 'relevant' edges (where the 'relevance' relies upon the INDVAL score) are considered as candidate information granules. Nonetheless, it is worth considering that the INDVAL score can be evaluated on ideally any subgraph, being not a peculiarity of single edges: indeed, one can also evaluate the INDVAL score on paths on any length within the graph (straightforwardly, an edge is a path of length 1). Amongst the two techniques, the former is easier to evaluate: the number of simplices is way lower than the number of paths. If the Clique Complex is used, then the worst-case complexity required in order to evaluate the simplicial complex starting from an n-vertex graph is $\mathcal{O}(3^{n/3})$ [280, 359]. Conversely, if the Vietoris-Rips is used, then one should build the Vietoris-Rips neighbourhood graph by scoring edges if nodes are at most ϵ apart and then evaluate the Clique Complex [410], augmenting the overall complexity to $\mathcal{O}(n^2) + \mathcal{O}(3^{n/3})$. The second technique, as instead, needs to evaluate all simple paths (e.g., using a Depth First Search-like algorithm [339]): given a graph $\mathcal{G}$ with n nodes and $|\mathcal{E}|$ edges, a single path can be found in $\mathcal{O}(n + |\mathcal{E}|)$, yet the number of simple paths goes like $\mathcal{O}(n!)$ in the worst-case scenario.

The four hypergraph kernels described in Section 4.5 also rely on simplicial complexes, so the computational complexity required for a single simplicial complex evaluation does not change with respect to the embedding via simplicial complexes case. However, the kernel matrix must be evaluated for each pair of simplicial complexes, adding an additional $\mathcal{O}(N^2)$ cost, if N is the number of patterns.

Finally, the multiple kernel approach described in Section 4.6 is the most expensive technique, computationally speaking. Indeed, for the $N_R \equiv N_K$ representations, the full dissimilarity space must be built, with $\mathcal{O}(N^2)$ cost. Fortunately, these dissimilarity space matrices must be evaluated only once. The main drawback is the tuning of the prototype selection and the kernel weights, which lead to many multiple kernel evaluation, each with cost $N_K \cdot \mathcal{O}(N^2)$.

Practically speaking, the multiple kernel technique is the only one in which hardware acceleration turned out to be crucial in order to ensure reasonable training times: radial basis kernels rely on pairwise Euclidean distances (see the exponential argument, Table 3.1), that along with other matrix operations such as sum and element-wise exponential can easily be parallelised on Graphics Processing Units (GPUs). To this end, a 12GB NVIDIA GeForce GTX TITAN X has been used, along with the MATLAB® Parallel Computing Toolbox GPU frontend. Nonetheless, the two embedding techniques and the two feature generation techniques can also be parallelised in order to deal with large datasets: distributed implementations are sketched in Appendix B.

Leaving the respective complexities aside, three techniques deserve an additional, interesting remark. The two embedding techniques (via simplicial complexes and INDVAL) and the multiple kernel approach allow an additional knowledge discovery phase, which cannot be carried out when using the other three techniques (spectral density, Betti numbers and hypergraph kernels). The embedding via simplicial complexes technique relies on representing each simplicial complex according to the histogram of simplices within the training data, so the following question might arise

> Instead of using all simplices drawn from the training data, there exist a subset of meaningful simplices? If so, can those simplices give the field-experts some further insights about the modelled system? Can they be human-interpretable?

The embedding via INDVAL follows pretty much the same strategy by representing each graph according to the histogram of relevant edges within the training data. So the very same questions can be asked.

The multiple kernel approach allows a two-fold knowledge discovery phase. Indeed,

1. by analysing the kernel weights (i.e., ω_i in Eq. (4.34)) one can determine which are the most important representations for solving the problem at hand (higher weights)

2. by analysing the prototype selector (i.e., $\mathbf{w}$ in Eq. (4.39)) one can analyse which are the patterns elected as pivotal for building the dissimilarity space.

Those questions will be answered in the next Chapter with the help of field-experts, where the INDVAL-based embedding will be tested on real metabolic pathways data and the other two techniques will be tested on real proteomic data.

Chapter 5

Tests and Results

5.1 Datasets Description

5.1.1 *E. coli* str. K12 PCN-EC

Since proteins perform a vast array of functions within living organisms, the target of this analysis is the prediction of enzymatic properties by relying on the Enzyme Commission (EC) nomenclature scheme [378]. The EC nomenclature classifies proteins (enzymes) according to the chemical reaction they catalyse by means of four digits separated by dots: the first digit discriminates amongst the six big enzymatic groups (see Table 5.1) and the latter three represent a progressively finer enzyme classification. In this work, the first digit is considered as the ground-truth class label for supervised machine learning purposes. However, since not all proteins are enzymes, a further class of 'not enzymes' is considered: this class indicates proteins with no enzymatic characteristics or proteins for which enzymatic characteristics are still unknown nowadays. It is worth noting that the data retrieval phase has been performed before August 2018, when the EC scheme introduced EC7 (Translocases).

TABLE 5.1: The six big EC nomenclature groups.

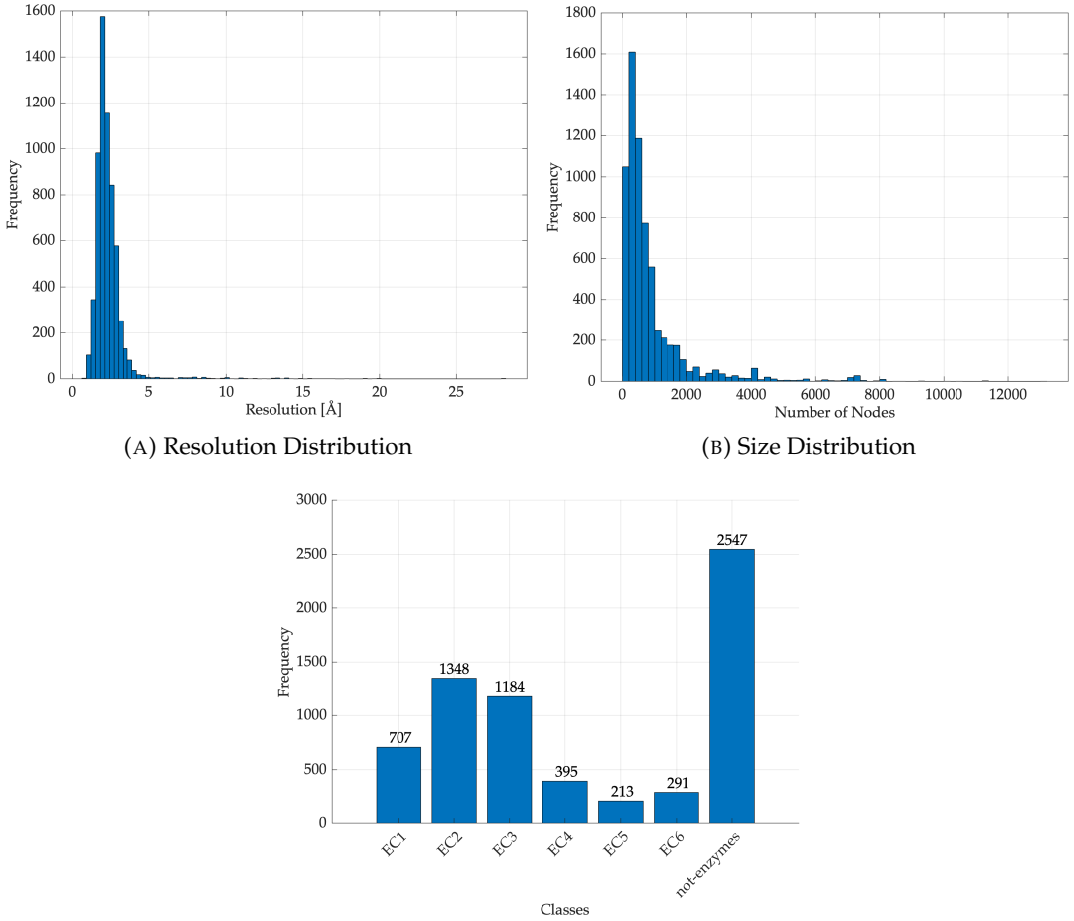
EC	Family	Reaction
1	Oxidoreductases	Transfer of electrons between two molecules (reductant and oxidant)
2	Transferases	Transfer of functional groups between two molecules (donor and acceptor)
3	Hydrolases	Breakage using water (hydrolysis) of a chemical bond
4	Lyases	Breakage of chemical bonds (no hydrolysis, no oxidation)
5	Isomerases	Isomer conversion, intermolecular arrangement
6	Ligases	Merging of two molecules

The dataset at the basis of the EC-related experiments has been constructed as follows:

1. on 8 April 2018 the entire *Escherichia coli* str K12 proteome has been retrieved from UniProt [355]
2. the list has been cross-checked with Protein Data Bank [38] in order to consider only resolved proteins (i.e., proteins whose 3D structure is available): for those proteins, the corresponding *.pdb* files have been collected
3. proteins with multiple EC numbers have been discarded
4. in *.pdb* files containing multiple structure models, only the first model is retained; similarly, for atoms having alternate coordinate locations, only the first location is retained.

Data collection and cleaning have been performed thanks to the following Python libraries: BioServices [79] for accessing online databases programmatically, BioPython [69, 77] and BioPandas [318] for manipulating and parsing *.pdb* files.

The dump leaded to a total number of 6685 proteins and some basic statistics are summarised in Figure 5.1. In order to preserve only good quality structures, by jointly considering the PCNs connectivity range (namely $[4, 8]$Å) and the resolution distribution in Figure 5.1a, all proteins with resolution greater than 3Å have been discarded and proteins with no information about the measurement resolution in the *.pdb* file header have been discarded as well. These filtering procedures dropped the number of available proteins from 6685 to 5583: hereinafter, this 5583-proteins dataset will be referred to as PCN-EC and serves a 'baseline dataset' for all experiments regarding the EC number prediction in order to make the results as comparable as possible, although minor filtering procedures will be performed on an experiment-related basis.



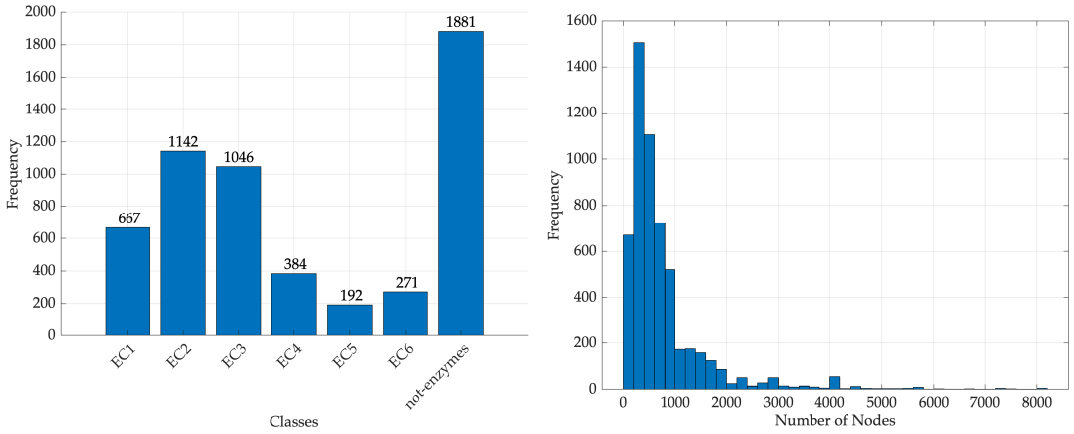
(A) Resolution Distribution

(B) Size Distribution

(C) Classes Distribution

FIGURE 5.1: Statistics for the initial 6685 PCNs dump (*E. coli* str. K12).

For the sake of completeness, Figure 5.2 shows some statistics about PCN-EC.



(A) Classes Distribution (B) Size Distribution

FIGURE 5.2: Statistics for PCN-EC.

5.1.2 *E. coli* str. K12 PCN-SOL

A second PCN-related problem consists in predicting proteins' solubility degree (folding propensity) starting from their PCN. Both problems are very hard to solve because no biochemical predictive theory is currently available. In the first problem (EC prediction), the difficulty resides in the fact that functional classification of enzymes is inherently sloppy: the chemical reactions defining the classes are largely superimposable and, in any case, mainly influence a minor part of the protein structure (active site). In this second problem (solubility/folding propensity prediction), the difficulty resides in the fact that the solubility of a protein *in vivo* is largely determined by the contemporary presence of other proteins that influence the folding of the target protein, while the specific reference database [293] is based on 'isolated proteins' *in vitro* (intrinsic solubility) and has to do with the folding prediction, that is still out of reach in protein science [295].

The data retrieval process can be summarised as follows:

1. from the eSOL database[29], containing the solubility degree (in percentage) for the *E. coli* proteins using the chaperone-free PURE system [344], the entire dump has been collected

2. proteins with no information about their solubility degree have been discarded

3. in order to enlarge the number of samples[30], we reversed the JW-to-PDB relation by downloading all structure files (if any) related to each JW entry from eSOL. Each structure will inherit the solubility degree from the JW entry

4. inconsistent data (e.g. same PDB with different solubility values) have been discarded; duplicates have been removed in case of redundant data (e.g., one solubility per PDB, but multiple JW's)

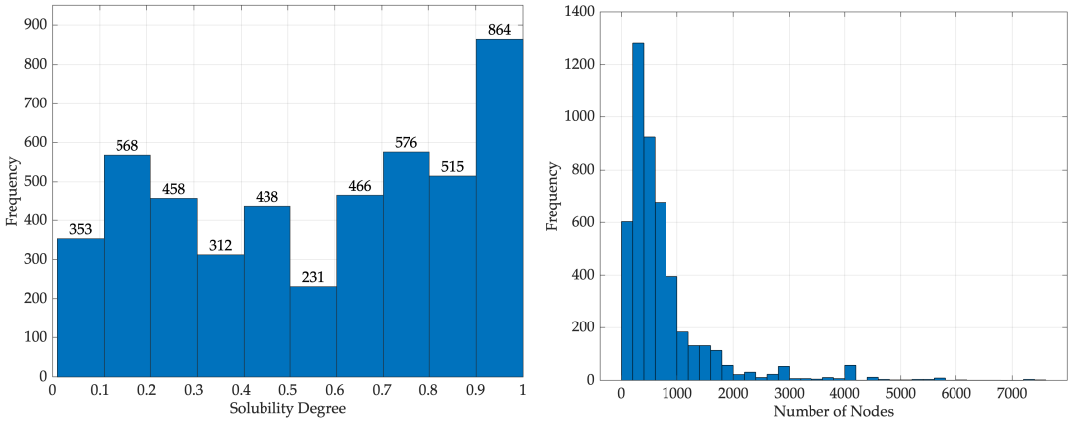
5. proteins having solubility degree greater than 100% have been set as 100%. The (small) deviations from 100% can be ascribed to minor experimental errors. After straightforward normalisation, the solubility degree can be considered as a real-valued number in range $[0, 1]$

[29] eSOL database (`http://tp-esol.genes.nig.ac.jp/`) developed in the Targeted Proteins Research Project.

[30] From the entire dump, only 432 proteins have their corresponding PDB identifier.

6. as per the PCN-EC dataset (Section 5.1.1), *.pdb* files have been parsed by removing alternate models and alternate atom locations.

The initial dump counted 5517 proteins and, following the same rationale as per Section 5.1.1, proteins with no information about measurement resolution and proteins with resolution greater than 3Å have been removed, leading to a final dataset (hereinafter PCN-SOL) composed by 4781 proteins. Basic statistics are summarised in Figure 5.3.



(A) Solubility Degree Distribution (B) Size Distribution

FIGURE 5.3: Statistics for PCN-SOL.

5.1.3 Metabolic Networks

Using the Python BioServices library, from the KEGG database [186–188], the entire dump of 5299 organisms whose metabolic network is known has been retrieved on 22-24 April 2018. The KEGG API returns data in an edge list-like format, starting from which the metabolic network can easily be built. Each organism has been described by its own metabolic network and mapped with a ground-truth class label according to the Linnaeus' taxonomical classification. Specifically, four classification tasks will be investigated:

Cell: The entire set of 5299 organisms has been divided in two classes, Eukaryotes and Prokaryotes, on the base of their basic cellular architecture. This is the most fundamental division of life located at the very basis of cellular architecture. As for the metabolic network wiring: the differences between prokaryotic (bacteria and archaea, the most simple forms of life) and eukaryotic (all the other organisms, from unicellular fungi to mammals) cell are striking.

Kingdom: The entire set of 5299 organisms has been divided in six classes, depending on their kingdom: Animals, Archaea, Bacteria, Fungi, Plants and Protists. This is mainly an 'exercise on evolution': the kingdom is the highest level of biological classification and the discrimination of different kingdoms by means of metabolic network wiring is a sort of benchmark for proposing metabolic networks as 'universal phenotype' for evolution studies. This is not a trivial task if one considers that while all the organisms can be compared as for their genotype (basically, DNA sequences), this does not hold for phenotypes

(i.e., actual physiological features). It is in fact very difficult to think of a sufficiently complex physiological process in common between a yeast and a dog other than the need to perform an efficient metabolism.

Animals: The third problem has to do with the classification of metazoa (multicellular animal species made of organised tissues and organs). In order to ensure a fair balance between number of classes and number of samples per class:

- organisms with no (Linnaeus') class have been discarded
- (Linnaeus') classes having less than 5 organisms have been discarded

The resulting dataset consists in 143 organisms, divided in 5 classes: Birds, Fishes, Insects, Mammals and Reptiles. The relevance of this exercise is linked to the rational choice of species whose metabolism is specifically suited for pharmacological or toxicological testing, see [198].

Bacteria: The fourth problem deals with classification of bacteria. Again, in order to ensure a fair balance between number of classes and number of samples per class: all (Linnaeus') classes having less than 50 organisms have been discarded. The resulting dataset consists in 1456 organisms divided in 17 classes: Bacillus, Bifidobacterium, Burkholderia, Campylobacter, Chlamydia, Clostridium, Corynebacterium, Escherichia, Helicobacter, Lactobacillus, Mycobacterium, Mycoplasma, Pseudomonas, Salmonella, Staphylococcus, Streptococcus, Streptomyces. The phenotypic classification of bacteria is a particularly hard problem [53] but has a crucial importance in biomedicine. Two applications with potentially huge consequences for human health are: a) the definition of 'healthy' and 'pathological' microbiota (the symbiotic/parasitic micro-organism communities playing a crucial role in human metabolism and disease, see [247]); b) the development of new antibiotic molecules efficiently contrasting emerging iatrogenic infections, see [380].

Figure 5.4 shows the classes distribution for the four problems.

5.1.4 Benchmark Datasets

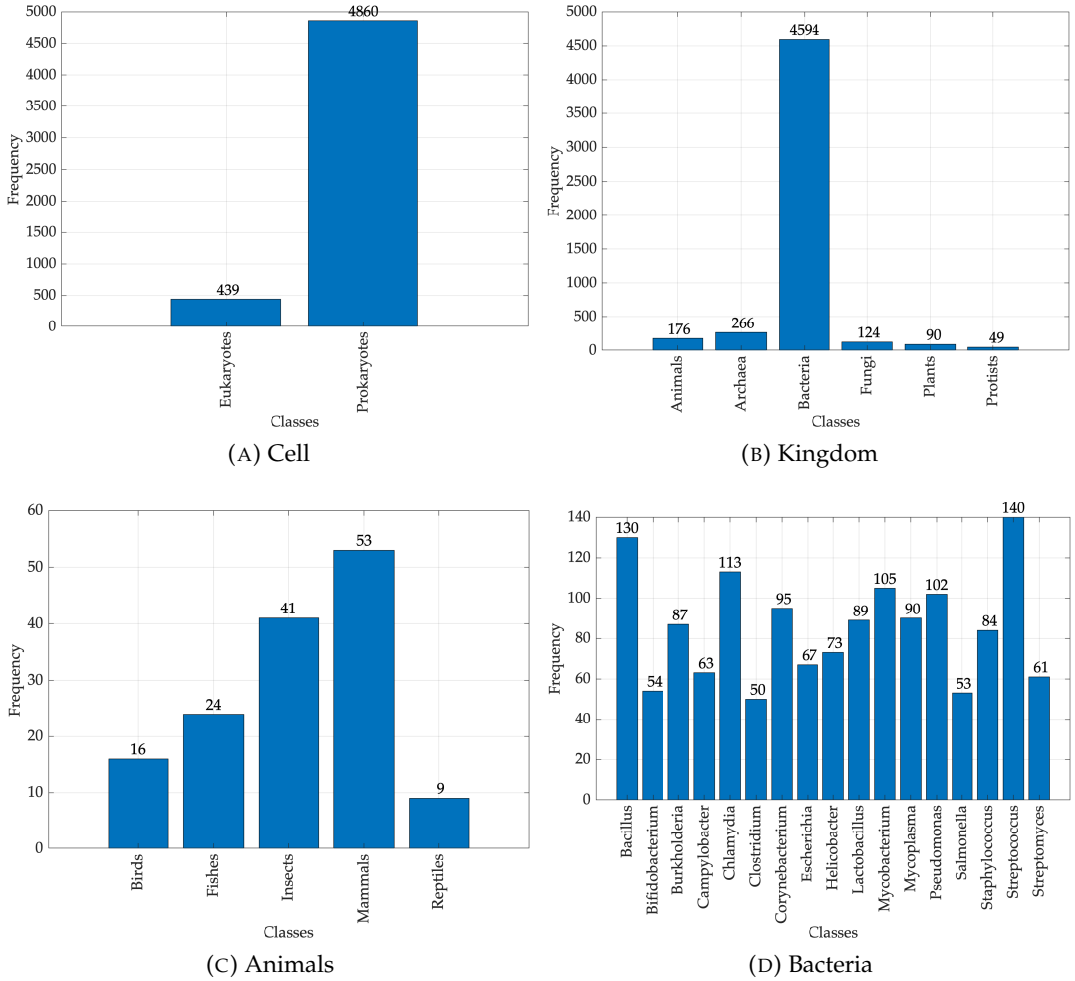
Alongside the two biological case studies, several of the proposed techniques from Chapter 4 will also be tested on well-known benchmark datasets, whose main characteristics are summarised in Table 5.2. All datasets have been downloaded from [195] and no pre-processing has been performed. All datasets are featured by unlabelled edges and nodes labelled with categorical attributes.

5.2 PCN Experiments: Enzymatic Properties

5.2.1 EC Classification via Spectral Density

The aim of this experiment is to address whether it is possible to predict the enzymatic function of a protein starting from its spectral density, following Section 4.1. Preliminary versions of this experiment can be found in [258] and [262].

From the baseline PCN-EC dataset (5583 proteins, see Section 5.1.1), all proteins having at least one isolated node have been discarded: indeed, such graphs will have at least one zero on the main diagonal of the degree matrix $\mathbf{D}$, making impossible the evaluation of $\mathbf{D}^{-1/2}$ (see Eq. (2.7)). This filtering procedure dropped the number of available proteins from 5583 to 5554.



(A) Cell (B) Kingdom

(C) Animals (D) Bacteria

FIGURE 5.4: Classes Distribution for the four Metabolic Network Datasets.

For each of the remaining proteins, the feature generation technique described in Section 4.1 and depicted in Figure 4.1, here summarised for the sake of ease, has been performed:

1. evaluation of the adjacency matrix $\mathbf{A}$ by scoring edges whether the Euclidean distance between residues' α-carbon atoms falls within $[4, 8]$Å
2. evaluation of the degree matrix $\mathbf{D}$, as in Eq. (2.2)
3. evaluation of the Laplacian matrix $\mathbf{L}$, as in Eq. (2.4)
4. evaluation of the normalised Laplacian matrix $\overline{\mathbf{L}}$, as in Eq. (2.7)
5. evaluation of the spectral decomposition of $\overline{\mathbf{L}}$, as in Eq. (2.8)
6. evaluation of the spectral density via kernel density estimator, as in Eq. (4.2), and final uniform sampling of $m = 100$ points in range $[0, 2]$.

The entire dataset (5554 patterns in $\mathbb{R}^{100}$) has been split into three non-overlapping Training, Validation and Test Set (50%, 25% and 25% of the total number of patterns, respectively) satisfying Eqs. (4.6)–(4.8): the splitting has been performed in a stratified fashion in order to preserve labels' distribution across the three subsets. The

TABLE 5.2: Benchmark Datasets for Graph Classification.

Dataset Name	# of patterns	# of classes	Avg. # of nodes	Avg. # of edges
AIDS	2000	2	15.69	16.2
BZR	405	2	35.75	38.36
COX2	467	2	41.22	43.45
DHFR	467	2	42.43	44.54
DD	1178	2	284.32	715.66
ENZYMES	600	6	32.63	62.14
FIRSTMM_DB	41	11	1377.27	3074.1
KKI	83	2	26.96	48.42
Mutagenicity	4337	2	30.32	30.77
MSRC_9	221	8	40.58	97.94
MSRC_21	563	20	77.52	198.32
MUTAG	188	2	17.93	19.79
NCI1	4110	2	29.87	32.3
NCI109	4127	2	29.68	32.13
OHSU	79	2	82.01	199.66
Peking_1	85	2	39.31	77.35
PTC_FM	349	2	14.11	14.48
PTC_FR	351	2	14.56	15
PTC_MM	336	2	13.97	14.32
PTC_MR	344	2	14.29	14.69
PROTEINS	1113	2	39.06	72.82
SYNTHETIC	300	2	100	196
Tox21_AHR	8169	2	18.09	18.5
Tox21_AR	9362	2	18.39	18.84
Tox21_AR-LBD	8599	2	17.77	18.16
Tox21_ARE	7167	2	16.28	16.52
Tox21_aromatase	7226	2	17.5	17.79
Tox21_ATAD5	9091	2	17.89	18.3
Tox21_ER	7697	2	17.58	17.94
Tox21_ER_LBD	8753	2	18.06	18.47
Tox21_HSE	8150	2	16.72	17.04
Tox21_MMP	7320	2	17.49	17.83
Tox21_p53	8634	2	17.79	18.19
Tox21_PPAR-gamma	8184	2	17.23	17.55

LibSVM implementation [67] of ν-SVM [336] has been used in order to perform classification[31]. The latter is equipped with the radial basis function kernel (cf. Table 3.1) of the form

$$K(\mathbf{a}, \mathbf{b}) = e^{-\gamma \cdot d^2(\mathbf{a},\mathbf{b},\mathbf{w})} \tag{5.1}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^m$ are two given patterns drawn from the dataset, γ is the kernel shape parameter and

$$d^2(\mathbf{a}, \mathbf{b}, \mathbf{w}) = \sum_{i=1}^{m} \mathbf{w}_i (\mathbf{a}_i - \mathbf{b}_i)^2 \tag{5.2}$$

[31] All experiments in this Chapter see SVMs as base classification system in order to stress the comparison between the proposed graph-based pattern recognition systems rather than classifiers as such. Preliminary tests shown that SVMs presented a good tradeoff between efficiency, effectiveness and ease of tuning with respect to other well-known classifiers such as Artificial Neural Networks, Naïve Bayes, Decision Trees and K-Nearest Neighbours (data not shown).

is the (squared) weighted Euclidean distance with $\mathbf{w} \in [0,1]^m$ acting as the weighting vector in order to investigate whether some samples (i.e., portions of the spectral density curve) are more important than others.

In order to automatically select in a data driven fashion not only suitable weights $\mathbf{w}$, but also the two SVM hyperparameters (i.e., the regularisation term ν and the kernel shape γ), a genetic algorithm [153] has been employed. Specifically, the genetic code has the form

$$\begin{bmatrix} \nu & \gamma & \mathbf{w} \end{bmatrix} \tag{5.3}$$

where $\nu \in (0,1]$ by definition, $\gamma \in (0,100]$ and $\mathbf{w} \in [0,1]^{100}$. The genetic optimisation aims at the maximisation of the informedness[32] J [395], defined as

$$J = \text{Sensitivity} + \text{Specificity} - 1 \in [-1,+1] \tag{5.4}$$

The model synthesis can be summarised as follows:

1. each individual from the evolving population strips the three parameters written in its genetic code, namely, ν, γ and $\mathbf{w}$
2. a ν-SVM which exploits γ and $\mathbf{w}$ in order to evaluate the kernel matrix (cf. Eqs (5.1)–(5.2)) and ν as regularisation parameter is trained on the Training Set
3. the fitness function (to be maximised) is the informedness as in Eq. (5.4) and is evaluated on the Validation Set
4. at the end of the evolution, the best individual is retained and finally tested on the Test Set.

The genetic algorithm has been configured to host 100 individuals with a maximum number of 100 generations; the elitism operator moves the best 10% of the individuals unaltered to the next generation; the crossover operator acts in a scattered fashion; the selection follows the roulette wheel heuristic; the mutation adds a random number drawn from a zero-mean Gaussian distribution whose variance shrinks as generations go by. A strict early stopping criterion halts the evolution if the shift in the fitness function for $1/3^{\text{rd}}$ of the generations is below 10^{-6}.

Table 5.3 shows the average results on the Test Set across five Training-Validation-Test splits in order to account the intrinsic randomness in the model synthesis. Results are presented via the following performance indices, expressed in percentage:

Accuracy: $\text{ACC} = \dfrac{TP + TN}{TP + TN + FP + FN}$

Sensitivity (also, Recall): $\text{SNS} = \dfrac{TP}{TP + FN}$

Specificity: $\text{SPC} = \dfrac{TN}{TN + FP}$

Positive Predictive Value (also, Precision): $\text{PPV} = \dfrac{TP}{TP + FP}$

[32] The rationale behind the choice of informedness follows: the informedness not only considers true positives, true negatives, false positives and false negatives simultaneously, but is also unbiased towards the most frequent class in case of heavily unbalanced problems. The same peculiarities do not hold in other most common performance indices such as the accuracy or the F-measure, as thoroughly investigated in [316].

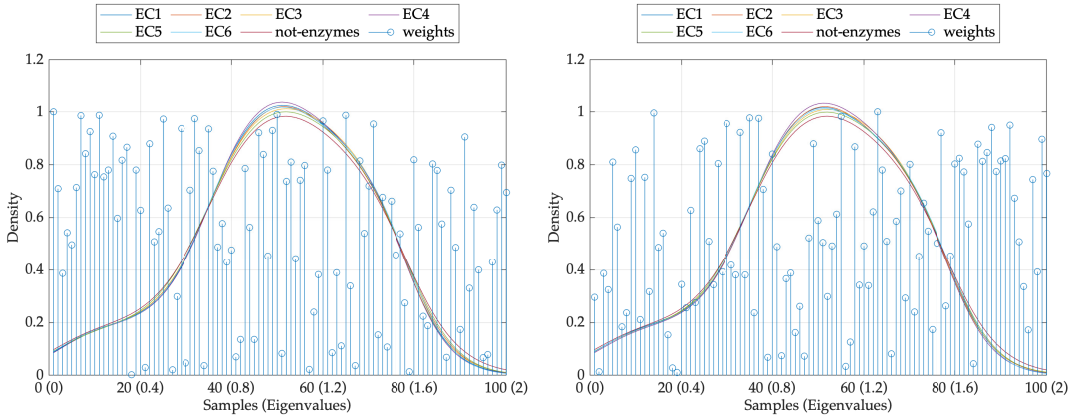
Negative Predictive Value: $\text{NPV} = \dfrac{TN}{TN + FN}$

where TP, TN, FP and FN indicate true positives, true negatives, false positives and false negatives, respectively.

TABLE 5.3: EC classification via spectral density, performances on the Test Set (avg $\pm$ std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	92.51 ± 0.42	47.12 ± 2.95	98.71 ± 0.57	83.73 ± 6.01	93.18 ± 0.33
EC2	86.77 ± 0.25	50.49 ± 2.18	96.11 ± 0.45	76.99 ± 1.50	88.30 ± 0.42
EC3	88.07 ± 0.74	49.50 ± 4.35	96.92 ± 0.41	78.66 ± 2.03	89.33 ± 0.81
EC4	94.67 ± 0.13	29.79 ± 2.61	99.38 ± 0.13	77.96 ± 3.37	95.12 ± 0.17
EC5	93.66 ± 7.01	17.50 ± 7.88	96.39 ± 7.53	55.45 ± 28.29	97.03 ± 0.08
EC6	96.10 ± 0.30	29.19 ± 4.85	99.53 ± 0.08	75.77 ± 6.01	96.48 ± 0.23
not-enzymes	79.39 ± 0.84	57.44 ± 2.29	90.57 ± 1.04	75.63 ± 1.88	80.71 ± 0.79

From Table 5.3 it is possible to see that all classifiers have satisfactory results, yet all of them share a rather low sensitivity (high false negatives). There are also minor false positives, as shown by precision (circa 70-80%). EC5 seems overall the most difficult to discriminate: sensitivity and precision are rather low with respect to the other classes.



(A) EC2 (B) EC4

FIGURE 5.5: EC classification via spectral density, weights vector for discriminating EC2 (left) and EC4 (right). Along with the weights values (blue stems), the average spectral density amongst the patterns belonging to the 7 classes are also shown.

Similarly, Figure 5.5 shows the resulting weights vector $\mathbf{w}$ as returned by the best amongst the five runs. For the sake of example, only the second (EC2) and forth classifiers (EC4) are shown in which it is possible to see that, for EC2, the rising front of the density curve (i.e., the first 20 samples, approximately) seems to be one of the most important parts for classification; conversely, for EC4, the falling front of the curve (i.e., the last 20 samples, approximately) seems to be one of the most important parts.

5.2.2 EC Classification via Betti Numbers

The aim of this experiment is to address whether it is possible to predict the enzymatic function of a protein starting from the sequence of Betti numbers at different

scales, following Section 4.2. Preliminary versions of this experiment can be found in [249].

Since the topological structure of a PCN is well-defined (nodes lie in a 3D space, edges exist whether the Euclidean distance between nodes is within $[4, 8]$Å), one might question the need of finding a suitable scale/connectivity resolution (e.g., ϵ for the Vietoris-Rips Complex) and argue that the Clique Complex is a suitable choice. So, given the Clique Complex, one can evaluate the three Betti numbers β_0, β_1 and β_2 and cast each PCN into a three-dimensional vector: this transformation has been performed on the entire PCN-EC dataset (5583 proteins). Since each PCN is cast in $\mathbb{R}^3$, Figure 5.6 shows the three-dimensional space spanned by β_0, β_1 and β_2 evaluated over the Clique Complex.

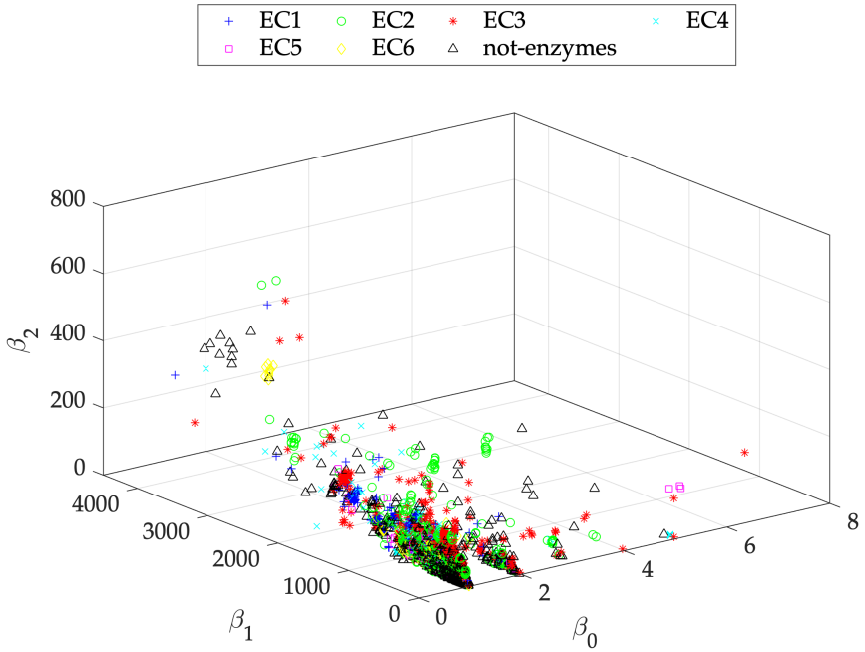


FIGURE 5.6: EC classification via Betti numbers, vector space using the Clique Complex.

By visual inspection, Figure 5.6 does not show enough separation amongst the seven classes, so one shall not expect interesting classification results. Nonetheless, after the usual Training-Validation-Test stratified split, ν-SVMs have been used as classification system where only ν and γ are tuned via genetic optimisation[33], following the same workflow as per the spectral density experiment discussed in Section 5.2.1. Table 5.4 shows the classification results (averaged across five Training-Validation-Test splits) when using the three Betti numbers evaluated over the Clique Complex: clearly, at least if compared with the spectral density experiment, there is a clear performance drop (especially as precision and sensitivity are concerned).

A second attempt sees the Vietoris-Rips Complex as core simplicial complex for the Betti numbers evaluation. Conversely to the Clique Complex, the Vietoris-Rips Complex needs a scale parameter ϵ, which can be addressed by considering the $[4, 8]$Å connectivity range. Specifically, five candidate values for ϵ have been considered, namely $\epsilon = \{4, 5, 6, 7, 8\}$ and, for each value of ϵ, the three Betti numbers (β_0, β_1 and β_2) have been evaluated over the resulting Vietoris-Rips Complex.

[33] In case of very few parameters, easier optimisation procedures can be considered in lieu of evolutionary metaheuristics, such as random search or grid search. Yet, for the sake of comparison, the very same genetic algorithm as per the previous Section has been employed for hyperparamters tuning.

TABLE 5.4: EC classification via Betti numbers (Clique Complex), performances on the Test Set (avg $\pm$ std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	81.98 ± 4.06	13.73 ± 9.25	91.20 ± 5.83	14.00 ± 7.93	88.69 ± 0.45
EC2	68.67 ± 4.85	37.82 ± 10.33	76.59 ± 8.52	30.14 ± 4.06	82.82 ± 1.06
EC3	83.83 ± 0.65	21.23 ± 2.86	98.24 ± 0.64	73.95 ± 6.94	84.42 ± 0.47
EC4	78.16 ± 14.95	21.04 ± 20.66	82.39 ± 17.54	8.51 ± 2.63	93.44 ± 0.44
EC5	61.88 ± 22.82	47.50 ± 22.17	62.39 ± 24.40	5.02 ± 1.64	97.11 ± 0.35
EC6	61.45 ± 6.72	56.42 ± 17.74	61.70 ± 7.74	6.81 ± 1.53	96.62 ± 0.95
not-enzymes	60.39 ± 2.08	60.17 ± 4.29	60.50 ± 5.06	43.74 ± 1.70	74.97 ± 0.85

Each protein from the baseline PCN-EC dataset has been cast towards a 15-length feature vector of the form

$$\left[\beta_0^{(4)}, \beta_1^{(4)}, \beta_2^{(4)}, \beta_0^{(5)}, \beta_1^{(5)}, \beta_2^{(5)}, \beta_0^{(6)}, \beta_1^{(6)}, \beta_2^{(6)}, \beta_0^{(7)}, \beta_1^{(7)}, \beta_2^{(7)}, \beta_0^{(8)}, \beta_1^{(8)}, \beta_2^{(8)}\right] \tag{5.5}$$

where, recall, $\beta_a^{(b)}$ indicates the a^{th} Betti number at $\epsilon = b$.

From the initial set of 5583 proteins composing PCN-EC, 9 proteins have been removed because out-of-memory errors have been thrown during the evaluation of the Betti numbers at $\epsilon = 8$, leading to a resulting set of 5574 proteins, which have been split in Training Set, Validation Set and Test Set. The experimental setup does not significantly change with respect to the spectral density case. The same genetic algorithm orchestrates the model synthesis using ν-SVMs equipped with the radial basis function kernel (Eq. (5.1)) whose core sees the weighted (squared) Euclidean distance from Eq. (5.2). Due to the different dimensionality of the feature vectors, for this experiment the weights vector reads as $\mathbf{w} \in [0, 1]^{15}$. The genetic algorithm setup remains unaltered.

TABLE 5.5: EC classification via Betti numbers (Vietoris-Rips Complex), performances on the Test Set (avg $\pm$ std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	89.83 ± 0.32	22.69 ± 4.37	98.96 ± 0.32	75.11 ± 3.46	90.41 ± 0.47
EC2	83.00 ± 0.50	26.25 ± 3.10	97.60 ± 0.64	74.02 ± 4.33	83.73 ± 0.53
EC3	84.92 ± 0.34	27.51 ± 2.82	98.16 ± 0.64	77.99 ± 4.59	85.45 ± 0.42
EC4	94.01 ± 0.18	18.33 ± 2.61	99.61 ± 0.13	78.07 ± 5.20	94.28 ± 0.17
EC5	85.34 ± 22.42	14.17 ± 27.06	87.88 ± 24.18	27.95 ± 48.04	96.68 ± 0.23
EC6	95.42 ± 0.21	10.07 ± 2.92	99.77 ± 0.11	69.44 ± 11.95	95.60 ± 0.17
not-enzymes	74.70 ± 0.58	41.32 ± 2.90	91.59 ± 0.86	71.35 ± 1.25	75.53 ± 0.78

Table 5.5 is the counterpart of Table 5.4 for the Vietoris-Rips Complex. The performance shifts between the Clique Complex and the Vietoris-Rips Complex are striking, with the latter greatly outperforming the former. Nonetheless, similar observations hold with respect to the spectral density experiment (Table 5.3): generally low sensitivity and EC5 still emerges as the most difficult class to discriminate.

Figure 5.7, finally, shows the best weights vector $\mathbf{w}$ amongst the five runs.

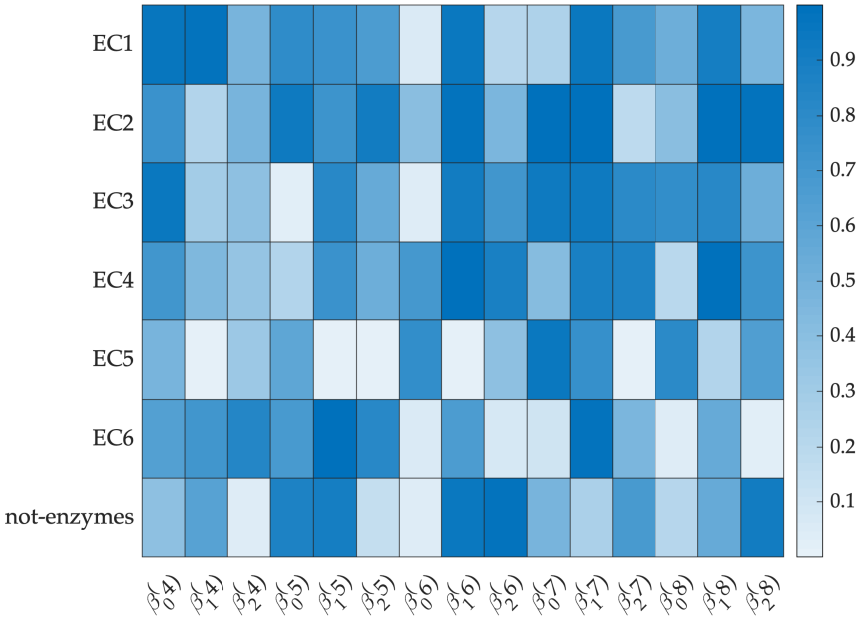


FIGURE 5.7: EC classification via Betti numbers, weights heatmap.

5.2.3 EC Classification by Embedding over Simplicial Complexes

For this experiment, discussed in [252], the entire PCN-EC dataset (5583 proteins) has been considered and properly split into Training, Validation and Test Set.

For each PCN within each set:

1. the Clique Complex has been considered as simplicial complex: indeed, the underlying 1-skeleton is already available thanks to existing knowledge on protein networks (i.e., edges are scored if nodes are within $[4, 8]$Å apart)

2. each node within each simplex has been identified by its node label (i.e., amino-acid type).

The alphabet $\mathcal{A}$ of candidate information granules is composed by the set of unique simplices by considering the simplicial complexes drawn from the Training Set and Validation Set. The unique simplices evaluation relies on the following two rules:

1. nodes ordering within the simplex has to be neglected: for example, the 2-simplex Arginine-Lysine-Serine and the 2-simplex Arginine-Serine-Lysine are considered equivalent. This can be easily done by properly sorting the node labels within each simplex, instead of trying out every possible permutation

2. due to the categorical nature of the node labels and as per the previous rule it is safe to say that two simplices are considered equal if they share the same order and they share the same node labels.

The alphabet synthesis returned a number of approximately 12000 symbols.

The next step is to build the symbolic histograms. For each simplicial complex within each set:

1. node labels are sorted within each simplex in order to facilitate the counting procedure with the alphabet symbols

2. each simplicial complex is cast towards a ≈ 12000 integer-valued vector which counts in position i the number of times the i^{th} alphabet symbol (a simplex) appears within the simplicial complex under analysis. The counting procedure still relies on the observation that two simplices are equal (i.e., increment counter) if they share the same order and the same node labels.

As the three instance matrices for Training, Validation and Test Set are build ($\mathbf{X}_{\text{TR}}$, $\mathbf{X}_{\text{VAL}}$ and $\mathbf{X}_{\text{TS}}$, respectively), the classification stage follows. For the sake of consistency with the two previous experiments, a radial basis function kernelised ν-SVM acts as classification system with the help of a genetic algorithm for hyperparameter tuning and feature selection. Indeed, by recalling Section 4.7, this experiment also aim at investigating the knowledge discovery peculiarity of information granulation-based pattern recognition systems. For a thorough and easy investigation, an important step is to shrink as much as possible the number of useful and pivotal symbols without loosing in classification performance.

The genetic code for each individual has the form

$$\begin{bmatrix} \nu & \gamma & \mathbf{w} \end{bmatrix} \tag{5.6}$$

where $\nu \in (0, 1]$ and $\gamma \in (0, 100]$ are the usual SVM hyperparameters and $\mathbf{w}$ is a boolean vector with as many elements as there are symbols in $\mathcal{A}$ (i.e., features in the instance matrices). The genetic algorithm setup does not change significantly with respect to the former two experiments, with the only difference being the mutation operator, which still follows a Gaussian behaviour for real-valued genes (ν and γ), but acts in a flip-the-bit fashion for boolean-valued genes (entries in $\mathbf{w}$).

The fitness function must be revisited as well in order to jointly consider the sparsity of the feature selector $\mathbf{w}$ and the classification performance. A suitable formulation reads as:

$$f = \alpha \cdot (1 - \bar{J}) + (1 - \alpha) \cdot \frac{|\mathbf{w} == 1|}{|\mathbf{w}|} \tag{5.7}$$

where the rightmost term takes into account the sparsity of the feature selector vector $\mathbf{w}$, $\bar{J}$ is the normalised[34] informedness, so the leftmost term takes into account the classification performances, and finally $\alpha \in [0, 1]$ is a user-defined parameter that weights the two contributions.

The model synthesis can be summarised as follows:

1. each individual from the evolving population strips the three parameters written in its genetic code, namely, ν, γ and $\mathbf{w}$
2. features corresponding to 1's in $\mathbf{w}$ are retained from $\mathbf{X}_{\text{TR}}$ and $\mathbf{X}_{\text{VAL}}$, whereas features corresponding to 0's are discarded
3. a ν-SVM which exploits γ in order to evaluate the kernel matrix (cf. Eqs (5.1)–(5.2)) and ν as regularisation parameter is trained on the shrunk version of $\mathbf{X}_{\text{TR}}$
4. the fitness function in Eq. (5.7) (to be minimised) is evaluated on the shrunk version of $\mathbf{X}_{\text{VAL}}$
5. at the end of the evolution, the best individual is retained: the $\mathbf{w}$ portion of its genetic code is used to shrink the Test Set and the final performances are evaluated.

[34] Recall from Section 5.2.1 that the informedness J is bounded in $[-1, +1]$. However, since the rightmost term is bounded in $[0, 1]$ a scaled version of the informedness has been used $\bar{J} = (J + 1)/2$ in order to be fairly combined.

For high-dimensional problems, evolutionary metaheuristics such as genetic algorithms can easily be trapped in local minima (maxima) due to the vast dimensionality of the search space, so one might wondering whether exists a smarter way of addressing the feature selection phase without optimising thousands of variables. A second classification system based on 1-norm SVMs (also ℓ_1-SVMs) [409] is proposed to address this problem. The rationale behind this choice follows:

- ℓ_1-SVMs minimise the 1-norm instead of the 2-norm of the separating hyperplane, as in 'standard' SVMs [50, 86, 90, 335, 336], so they return a solution (hyperplane coefficient vector) which is sparse: this leads to an intrinsic feature selection during training [409]
- it has been experimentally demonstrated that for high-dimensional and possibly sparse data (e.g., text classification, sentiment analysis) linear methods (a family which includes ℓ_1-SVMs) tend to have similar performances with non-linear methods, at a very small fraction of the training time [135]
- there exist efficient implementations for high-dimensional and sparse data, e.g. the LibLINEAR library [135] developed by the same team behind the already-mentioned LibSVM.

Since the feature selection phase is automatically performed during training, returning the hyperplane coefficient vector $\boldsymbol{\beta}$ which is natively sparse, the model synthesis must be revisited. For ℓ_1-SVMs the genetic code has the form:

$$\begin{bmatrix} C & \mathbf{c} \end{bmatrix} \tag{5.8}$$

where $C \in [-10, +10]$ is the regularisation term and $\mathbf{c}$ is a real-valued vector with entries also in $[-10, +10]$ which weights C in a class-wise fashion: for the i^{th} class, misclassifications are weighted as $C \cdot \mathbf{c}_i$. It is worth noting that these additional weights are not mandatory for ℓ_1-SVMs to work, but they might be helpful in case of unbalanced classification problems. The genetic optimisation does not change with respect to the previous case, other than the mutation rate (which works in a fully Gaussian fashion since there are no boolean genes) and the fitness function (which still reads as Eq. (5.7), but with $\boldsymbol{\beta}$ in lieu of $\mathbf{w}$).

Tables 5.6 and 5.7 show the classification performances (averaged across five Training-Validation-Test Set splits) for ν-SVMs and ℓ_1-SVMs on PCN-EC for $\alpha = 1$ and $\alpha = 0.5$ in the fitness function: in the former case, each classifier can choose as many features as required in order to maximise performances; in the latter case, the sparsity is equally considered along with performances. Alongside the five well-known performance indices (ACC, SNS, SPC, NPV, PPV), the *sparsity* is also shown, defined as the percentage of selected symbols: conversely to the performance indices, a lower value is preferred.

As ℓ_1-SVMs are concerned, by matching Tables 5.6 and 5.7 (i.e., when switching from $\alpha = 1$ to $\alpha = 0.5$), it is possible to see that ℓ_1-SVMs, other than selecting a smaller number of symbols, tend to improve in terms of SNS and NPV for almost all classes. Similarly, as regards ν-SVMs, they mainly benefit in terms of feature selection, with only class 7 showing minor performance improvements in terms of SNS and NPV.

TABLE 5.6: EC classification via embedding over simplicial complexes, performances on the Test Set at $\alpha = 1$ (avg $\pm$ std). In bold, the best between the two classification systems.

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
EC1	ν-SVM	**97.2 $\pm$ 0.4**	81.9 $\pm$ 2.7	**99.2 $\pm$ 0.3**	97.6 $\pm$ 0.3	**93.6 $\pm$ 2.4**	**11.5 $\pm$ 5.8**
	ℓ_1-SVM	97.0 $\pm$ 0.3	**87.3 $\pm$ 2.0**	98.4 $\pm$ 0.4	**98.3 $\pm$ 0.3**	88.0 $\pm$ 2.6	16.0 $\pm$ 10.0
EC2	ν-SVM	**94.5 $\pm$ 0.9**	79.7 $\pm$ 3.4	**98.3 $\pm$ 0.8**	95.0 $\pm$ 0.8	**92.2 $\pm$ 3.3**	26.1 $\pm$ 18.5
	ℓ_1-SVM	**94.5 $\pm$ 0.9**	**83.8 $\pm$ 2.7**	97.3 $\pm$ 0.5	**95.9 $\pm$ 0.7**	88.9 $\pm$ 2.0	**22.7 $\pm$ 11.4**
EC3	ν-SVM	**94.7 $\pm$ 0.7**	78.2 $\pm$ 1.9	**98.5 $\pm$ 0.5**	95.1 $\pm$ 0.4	**92.3 $\pm$ 2.5**	18.6 $\pm$ 8.1
	ℓ_1-SVM	93.3 $\pm$ 1.5	**83.9 $\pm$ 3.7**	95.5 $\pm$ 2.5	**96.3 $\pm$ 0.8**	81.9 $\pm$ 7.6	**17.4 $\pm$ 8.2**
EC4	ν-SVM	**97.3$\pm$ 0.4**	69.2 $\pm$ 8.5	**99.4 $\pm$ 0.3**	97.8 $\pm$ 0.6	**89.9 $\pm$ 3.8**	19.1 $\pm$ 16.6
	ℓ_1-SVM	**97.3 $\pm$ 0.4**	**79.8 $\pm$ 2.7**	98.6 $\pm$ 0.5	**98.5 $\pm$ 0.2**	81.6 $\pm$ 5.1	**7.8 $\pm$ 1.6**
EC5	ν-SVM	**98.7$\pm$ 0.2**	63.8 $\pm$ 7.3	**99.9 $\pm$ 0.1**	98.7 $\pm$ 0.3	**97.1 $\pm$ 3.4**	31.7 $\pm$ 19.5
	ℓ_1-SVM	97.9 $\pm$ 1.2	**70.4 $\pm$ 9.7**	98.9 $\pm$ 1.3	**98.9 $\pm$ 0.3**	75.3 $\pm$ 20.3	**5.1 $\pm$ 3.2**
EC6	ν-SVM	**99.1$\pm$ 0.6**	83.5 $\pm$ 10.3	**99.9 $\pm$ 0.1**	99.2 $\pm$ 0.5	**97.2 $\pm$ 2.6**	28.7 $\pm$ 14.4
	ℓ_1-SVM	98.7 $\pm$ 0.6	**86.2 $\pm$ 5.1**	99.4 $\pm$ 0.4	**99.3 $\pm$ 0.3**	87.8 $\pm$ 7.7	**9.6 $\pm$ 4.7**
not-enzymes	ν-SVM	87.4 $\pm$ 1.0	75.7 $\pm$ 2.3	**93.4 $\pm$ 0.7**	88.3 $\pm$ 1.0	**85.3 $\pm$ 1.5**	**6.9 $\pm$ 1.6**
	ℓ_1-SVM	**88.8 $\pm$ 1.4**	**83.4 $\pm$ 2.0**	91.6 $\pm$ 2.0	**91.6 $\pm$ 0.9**	83.6 $\pm$ 3.3	36.3 $\pm$ 13.2

TABLE 5.7: EC classification via embedding over simplicial complexes, performances on the Test Set at $\alpha = 0.5$ (avg $\pm$ std). In bold, the best between the two classification systems.

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
EC1	ν-SVM	**96.8 $\pm$ 0.4**	80.4 $\pm$ 3.7	**99.0 $\pm$ 0.5**	97.4 $\pm$ 0.5	**92.0 $\pm$ 3.9**	9.9 $\pm$ 6.5
	ℓ_1-SVM	95.3 $\pm$ 1.3	**87.1 $\pm$ 1.8**	96.5 $\pm$ 1.6	**98.2 $\pm$ 0.2**	77.7 $\pm$ 7.4	**3.3 $\pm$ 0.4**
EC2	ν-SVM	**93.9 $\pm$ 1.1**	77.8 $\pm$ 4.6	**98.0 $\pm$ 0.3**	94.5 $\pm$ 1.1	**90.9 $\pm$ 1.5**	6.8 $\pm$ 3.0
	ℓ_1-SVM	92.7 $\pm$ 1.1	**86.7 $\pm$ 3.6**	94.2 $\pm$ 1.4	**96.5 $\pm$ 0.9**	79.6 $\pm$ 3.8	**4.5 $\pm$ 0.3**
EC3	ν-SVM	**94.0 $\pm$ 0.6**	74.3 $\pm$ 2.8	**98.5 $\pm$ 0.2**	94.3 $\pm$ 0.6	**92.1 $\pm$ 1.3**	6.8 $\pm$ 5.3
	ℓ_1-SVM	92.1 $\pm$ 0.9	**84.4 $\pm$ 3.3**	93.8 $\pm$ 1.4	**96.3 $\pm$ 0.7**	76.2 $\pm$ 3.6	**4.0 $\pm$ 0.5**
EC4	ν-SVM	**97.3 $\pm$ 0.5**	69.6 $\pm$ 8.2	**99.3 $\pm$ 0.3**	97.8 $\pm$ 0.6	**88.4 $\pm$ 4.1**	12.8 $\pm$ 14.8
	ℓ_1-SVM	96.6 $\pm$ 0.4	**82.5 $\pm$ 3.6**	97.7 $\pm$ 0.7	**98.7 $\pm$ 0.3**	72.9 $\pm$ 5.3	**2.8 $\pm$ 0.4**
EC5	ν-SVM	**98.5 $\pm$ 0.3**	61.3 $\pm$ 11.3	**99.8 $\pm$ 0.1**	98.6 $\pm$ 0.4	**93.0 $\pm$ 4.2**	13.6 $\pm$ 11.9
	ℓ_1-SVM	96.9 $\pm$ 1.2	**71.7 $\pm$ 10.1**	97.8 $\pm$ 1.3	**99.0 $\pm$ 0.4**	56.9 $\pm$ 12.7	**1.8 $\pm$ 0.6**
EC6	ν-SVM	**98.9 $\pm$ 0.4**	80.3 $\pm$ 5.6	**99.9 $\pm$ 0.1**	99.0 $\pm$ 0.3	**97.1 $\pm$ 2.9**	23.3 $\pm$ 18.5
	ℓ_1-SVM	97.5 $\pm$ 1.7	**88.8 $\pm$ 2.5**	97.9 $\pm$ 1.7	**99.4 $\pm$ 0.1**	71.5 $\pm$ 15.1	**2.2 $\pm$ 0.3**
not-enzymes	ν-SVM	**87.4 $\pm$ 0.8**	76.5 $\pm$ 2.2	**93.0 $\pm$ 0.5**	88.6 $\pm$ 1.0	**84.7 $\pm$ 1**	6.5 $\pm$ 2.1
	ℓ_1-SVM	86.6 $\pm$ 1.1	**80.1$\pm$ 3.9**	89.9 $\pm$ 3.3	**89.9 $\pm$ 1.5**	80.5 $\pm$ 4.5	**4.8 $\pm$ 0.9**

By comparing the two classification systems it is possible to draw the following conclusions:

- at $\alpha = 1$: ℓ_1-SVMs outperform the kernelised counterpart in terms of SNS (all classes) and NPV (all classes), whereas ν-SVMs outperform the former in terms of SPC (all classes) and PPV (all classes). The overall ACC sees ℓ_1-SVMs outperforming ν-SVMs only for class 7, the two classifiers perform equally for classes 2 and 4 and for the remaining classes ν-SVMs perform better. Regardless of which performs the best in an absolute manner, the performance shifts are rather small as ACC, SPC and NPV are concerned ($\approx$ 3.3% or less), whereas interesting shifts include SNS (ℓ_1-SVMs outperforming by $\approx$ 10% on class 4) and PPV (ν-SVMs outperforming by $\approx$ 10% on class 3 and $\approx$ 22% on class 5);
- at $\alpha = 0.5$: ℓ_1-SVMs outperform the kernelised counterpart in terms of SNS (all classes) and NPV (all classes), whereas ν-SVMs outperform the former in terms of SPC (all classes), PPV (all classes) and ACC (all classes). While the performance shifts are rather small for ACC ($\approx 1-2\%$) and SPC ($\approx 3-4\%$), remarkable shifts regard PPV (ν-SVMs outperform up to 36% for class 5) and SNS (ℓ_1-SVMs outperform up to 13% for class 4).

Conversely, as the sparsity is concerned:

- at $\alpha = 1$: ℓ_1-SVMs select fewer symbols with respect to ν-SVMs only for classes 1 and 7
- at $\alpha = 0.5$: ℓ_1-SVMs outperform ν-SVMs for all classes.

Another non negligible aspect of this procedure is the time effort for the model synthesis due to the high dimensionality of the dataset at hand: ℓ_1-SVMs are incredibly faster with respect to the 2-norm counterpart, mainly for the following reasons:

- there is no need to explicitly evaluate the kernel matrix (see Eqs. (5.1)–(5.2)): ℓ_1-SVMs are linear classifiers by definition
- the training phase for ℓ_1-SVMs relies on solving a linear programming problem, whereas ν-SVMs solve a quadratic programming problem
- they automatically return a sparse solution, there is no need to optimise any feature selector[35].

5.2.4 EC Classification using Hypergraph Kernels

As per the previous experiment, which also regarded simplicial complexes, the entire PCN-EC dataset has been considered and properly split into Training, Validation and Test Set.

For each PCN within each set:

1. the Clique Complex has been evaluated
2. each vertex within each simplex has been identified by its own node label (i.e., amino-acid type).

All of the four kernels from Section 4.5 have been considered as prospective kernels for the ν-SVM. Recalling that the four kernels are parameter-free, as the Clique Complex does not depend on any scale parameter, the only parameter to be tuned is the regularisation term ν. This single variable optimisation has been performed via random search in the admissible domain $(0, 1]$: candidate values are tested on the Training Set and validated on the Validation Set. The value which maximises the informedness J on the Validation Set is retained and the final performances are evaluated on the Test Set.

Tables 5.8–5.11 show the average results across five Training-Validation-Test Set splits. For the EC number prediction, there is no kernel that significantly outperforms the others. However, like in the embedding over simplicial complexes case (Tables 5.6–5.7), the overall performances feature a remarkable boost with respect to the Betti numbers and the spectral density experiments.

5.2.5 EC Classification using Multiple Kernel Machines

For this experiment, discussed in [261], $N_K \equiv N_R = 8$ representations for each PCN have been considered:

[35] In this test a genetic algorithm for hyper-parameter optimisation has been used because of consistency with the ν-SVMs case. As already discussed in Section 5.2.2, since the number of hyper-parameters to be optimised is rather small (see Eq. (5.8)), lighter procedures can be employed instead.

TABLE 5.8: EC classification via Hypergraph Kernels, performance on the Test Set for Histogram Kernel (avg ± std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	97.76 ± 0.39	86.75 ± 3.74	99.25 ± 0.32	94.07 ± 2.26	98.23 ± 0.49
EC2	95.50 ± 0.43	84.84 ± 1.91	98.23 ± 0.29	92.51 ± 1.14	96.19 ± 0.46
EC3	95.17 ± 0.93	82.99 ± 1.85	97.97 ± 1.01	90.55 ± 4.17	96.16 ± 0.41
EC4	98.31 ± 0.46	81.46 ± 4.19	99.55 ± 0.26	93.14 ± 4.02	98.64 ± 0.31
EC5	98.92 ± 0.25	71.67 ± 8.01	99.90 ± 0.12	96.39 ± 4.37	99.00 ± 0.28
EC6	99.54 ± 0.12	91.04 ± 1.83	99.97 ± 0.04	99.34 ± 0.90	99.55 ± 0.09
not-enzymes	89.78 ± 0.57	81.91 ± 3.12	93.77 ± 1.53	87.10 ± 2.39	91.1 ± 1.32

TABLE 5.9: EC classification via Hypergraph Kernels, performance on the Test Set for Jaccard Kernel (avg ± std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	97.81 ± 0.39	84.70 ± 2.51	99.58 ± 0.21	96.45 ± 1.73	97.97 ± 0.33
EC2	95.93 ± 0.21	83.44 ± 1.73	99.14 ± 0.44	96.17 ± 1.85	95.89 ± 0.40
EC3	95.86 ± 0.39	80.69 ± 2.87	99.35 ± 0.37	96.66 ± 1.70	95.72 ± 0.61
EC4	98.16 ± 0.46	77.50 ± 5.97	99.69 ± 0.17	94.92 ± 2.93	98.36 ± 0.43
EC5	98.85 ± 0.36	69.58 ± 11.08	99.90 ± 0.12	96.33 ± 4.50	98.93 ± 0.39
EC6	99.46 ± 0.20	88.96 ± 3.75	99.98 ± 0.03	99.66 ± 0.77	99.45 ± 0.19
not-enzymes	90.42 ± 1.00	82.77 ± 2.54	94.31 ± 1.34	88.15 ± 2.41	91.52 ± 1.13

TABLE 5.10: EC classification via Hypergraph Kernels, performance on the Test Set for Edit Kernel (avg ± std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	97.62 ± 0.39	83.13 ± 3.24	99.58 ± 0.21	96.4 ± 1.71	97.76 ± 0.42
EC2	95.46 ± 0.25	81.89 ± 1.96	98.94 ± 0.61	95.28 ± 2.51	95.52 ± 0.44
EC3	95.54 ± 0.34	79.54 ± 1.73	99.22 ± 0.38	95.97 ± 1.88	95.47 ± 0.36
EC4	97.89 ± 0.57	73.75 ± 7.23	99.68 ± 0.18	94.37 ± 3.12	98.09 ± 0.52
EC5	98.72 ± 0.4	66.25 ± 13.04	99.88 ± 0.13	95.78 ± 4.79	98.81 ± 0.45
EC6	99.41 ± 0.06	88.06 ± 1.06	99.98 ± 0.03	99.67 ± 0.75	99.4 ± 0.05
not-enzymes	90.01 ± 0.77	82.34 ± 2.58	93.9 ± 0.97	87.32 ± 1.64	91.29 ± 1.15

TABLE 5.11: EC classification via Hypergraph Kernels, performance on the Test Set for Stratified Edit Kernel (avg ± std).

Class	ACC	SNS	SPC	PPV	NPV
EC1	97.53 ± 0.41	82.53 ± 3.27	99.56 ± 0.22	96.24 ± 1.81	97.69 ± 0.42
EC2	95.53 ± 0.17	82.04 ± 2.15	98.99 ± 0.37	95.48 ± 1.48	95.55 ± 0.49
EC3	95.67 ± 0.36	79.62 ± 2.05	99.37 ± 0.31	96.68 ± 1.57	95.49 ± 0.43
EC4	97.99 ± 0.46	75.42 ± 6.10	99.66 ± 0.17	94.30 ± 2.81	98.21 ± 0.44
EC5	98.74 ± 0.29	67.50 ± 9.03	99.85 ± 0.19	94.83 ± 6.52	98.85 ± 0.31
EC6	99.38 ± 0.20	87.46 ± 3.89	99.98 ± 0.03	99.66 ± 0.77	99.37 ± 0.19
not-enzymes	90.42 ± 0.72	82.77 ± 2.44	94.31 ± 0.84	88.12 ± 1.4	91.52 ± 1.07

1. Betti numbers sequence evaluated over the Vietoris-Rips Complex: 15-length integer-valued vector, as described in Section 5.2.2

2. Centrality measures: 27-length real-valued vector containing the degree centrality [289], the eigenvector centrality [289], the PageRank centrality [289], the Katz centrality [190, 289], the closeness centrality [289], the betweenness centrality [289], the edge betweenness centrality [52], the load centrality [52, 152], the edge load centrality, the subgraph centrality [134], the Estrada index [131], the harmonic centrality [47], the global reaching centrality [279], the average clustering coefficient [333], the average neighbour degree [31]. Whilst the average clustering coefficient, the Estrada index and the global reaching centrality are global characteristics (i.e., related to the whole graph), the others are local characteristics (i.e., related to individual nodes or edges): for these centrality measures, the mean and standard deviation across nodes or edges have been considered

3. Energy and Laplacian Energy: 2-length real-valued vector containing the energy and the Laplacian energy of a graph (see Eqs. (2.5) and (2.6))

4. Nodes Functional Cartography: 8-length real-valued vector containing the percentage of nodes belonging to each functional role[36] and the graph modularity according to the node classification by Guimerà and Amaral [161]

5. Heat Content Invariant: 4-length real-valued vector containing the first four heat content invariants (q_0, q_1, q_2, q_3) evaluated as in Eq. (2.17)

6. Heat Kernel Trace: 10-length real-valued vector containing the heat kernel trace at time instants $t = 1, 2, \ldots, 10$ evaluated as in Eq. (2.12)

7. Size: 4-length real-valued vector containing the number of nodes, the number of edges, the number of chains and the radius of gyration[37] [236]

8. Normalised Laplacian Spectral Density: 100-length real-valued vector, as described in Section 5.2.1.

For each of these eight representations, the full dissimilarity matrices have been built (Section 3.3.3) by using the Euclidean distance as dissimilarity measure. The genetic algorithm orchestrating the overall synthesis has the following genetic code

$$\begin{bmatrix} \nu & \boldsymbol{\gamma} & \boldsymbol{\omega} & \mathbf{w} \end{bmatrix} \tag{5.9}$$

where $\nu \in (0, 1]$ is the usual regularisation term for ν-SVMs, $\boldsymbol{\gamma} \in (0, 100]^8$ is an 8-length vector containing the kernel shapes (see Eq. (4.35)), $\boldsymbol{\omega}$ is an 8-length vector containing the kernel weights (see Eqs. (4.36)–(4.37)) and $\mathbf{w}$ is a boolean vector containing as many entries as there are training patterns in order to select suitable prototypes in the dissimilarity space. The genetic algorithm setup remains unchanged with respect to the embedding over simplicial complexes case (Section 5.2.3) and the model synthesis can be summarised as follows:

1. each individual from the evolving population strips the four parameters written in its genetic code, namely, ν, $\boldsymbol{\gamma}$, $\boldsymbol{\omega}$ and $\mathbf{w}$

2. from each of the 8 dissimilarity space matrices, the sub-matrices corresponding to the pairwise dissimilarities between training patterns are sliced (cf. Eq. (4.38))

[36] Non-hub nodes: ultra-peripherals, peripherals, non-hub connectors, non-hub kinless.
Hub nodes: provincial hubs, connector hubs, kinless hubs.

[37] The radius of gyration measures the compactness of the folded protein structure with respect to its centre of mass.

3. columns (prototypes) corresponding to 0's in $\mathbf{w}$ are discarded (see Eq. (4.39))
4. eight kernels with shapes γ are individually evaluated over the eight (reduced) dissimilarity matrices and combined in a single kernel matrix thanks to ω (see Eq. (4.40))
5. a ν-SVM which exploits the previously-evaluated kernel matrix and the regularisation term ν is trained
6. the fitness function in Eq. (5.7) (to be minimised) is evaluated on the Validation Set (i.e., steps 3 and 4 are repeated by considering the pairwise dissimilarities between validation and training patterns and fed to the SVM trained in step 5)
7. at the end of the evolution, the best individual is retained and the final performances are evaluated on the Test Set (i.e., steps 3 and 4 are repeated by considering the pairwise dissimilarities between test and training patterns and fed to the SVM trained on the Training Set with the best parameters).

From the initial PCN-EC dataset (5583 patterns), as in Section 5.2.2, 9 proteins had to be removed because of the impossibility to evaluate the Betti numbers at $\epsilon = 8$.

Table 5.12 shows the average results on the Test Set across five Training-Validation-Test splits. As per the embedding over simplicial complexes experiment (Section 5.2.3), the sparsity of the feature selector in the dissimilarity space is also shown. Further, both $\alpha = 1$ and $\alpha = 0.5$ have been considered in the fitness function (see Eq. (5.7)). Finally, Figures 5.8 and 5.9 show the best kernel weights ω amongst the five runs for each class at $\alpha = 1$ and $\alpha = 0.5$, respectively. The performances, especially in terms of sensitivity, seems to be lower with respect to the two simplicial complexes-based techniques from Sections 5.2.3 and 5.2.4. Nonetheless, the multiple kernel technique outperforms both the Betti numbers technique (Section 5.2.2) and the spectral density technique (Section 5.2.1).

TABLE 5.12: EC classification via multiple kernel machines, performances on the Test Set (avg $\pm$ std). In bold, the best between the α values.

Class	α	ACC	SNS	SPC	PPV	NPV	Sparsity
EC1	0.5	94.47 $\pm$ 0.61	63.46 $\pm$ 3.73	98.68 $\pm$ 0.41	86.77 $\pm$ 3.66	95.22 $\pm$ 0.47	**4.50 $\pm$ 2.22**
	1	**94.69 $\pm$ 0.59**	**64.54 $\pm$ 3.63**	**98.78 $\pm$ 0.39**	**87.81 $\pm$ 3.54**	**95.36 $\pm$ 0.45**	45.81 $\pm$ 23.13
EC2	0.5	90.51 $\pm$ 1.24	65.96 $\pm$ 4.30	96.82 $\pm$ 0.65	84.21 $\pm$ 3.24	91.71 $\pm$ 0.99	**3.58 $\pm$ 0.49**
	1	**91.06 $\pm$ 0.94**	**67.58 $\pm$ 3.84**	**97.09 $\pm$ 0.96**	**85.84 $\pm$ 3.83**	**92.10 $\pm$ 0.85**	38.27 $\pm$ 11.10
EC3	0.5	90.73 $\pm$ 0.47	**60.54 $\pm$ 2.67**	97.69 $\pm$ 0.38	85.82 $\pm$ 1.86	**91.48 $\pm$ 0.52**	**3.61 $\pm$ 0.43**
	1	**90.78 $\pm$ 0.79**	60.46 $\pm$ 2.75	**97.77 $\pm$ 0.57**	**86.27 $\pm$ 3.24**	91.47 $\pm$ 0.56	55.93 $\pm$ 11.85
EC4	0.5	**96.34 $\pm$ 0.36**	**54.37 $\pm$ 4.50**	**99.44 $\pm$ 0.06**	**87.79 $\pm$ 2.06**	**96.72 $\pm$ 0.32**	**7.82 $\pm$ 4.93**
	1	96.22 $\pm$ 0.35	53.33 $\pm$ 3.41	99.4 $\pm$ 0.24	86.91 $\pm$ 4.9	96.64 $\pm$ 0.24	29.14 $\pm$ 15.03
EC5	0.5	97.72 $\pm$ 0.16	37.92 $\pm$ 4.27	**99.85 $\pm$ 0.05**	**90.09 $\pm$ 3.14**	97.83 $\pm$ 0.15	**8.78 $\pm$ 4.31**
	1	**97.73 $\pm$ 0.28**	**38.33 $\pm$ 7.00**	**99.85 $\pm$ 0.09**	90.04 $\pm$ 6.00	**97.84 $\pm$ 0.24**	44.64 $\pm$ 9.87
EC6	0.5	97.86 $\pm$ 0.35	62.65 $\pm$ 7.56	99.65 $\pm$ 0.21	90.53 $\pm$ 5.46	98.13 $\pm$ 0.38	**5.95 $\pm$ 4.30**
	1	**97.95 $\pm$ 0.21**	**63.53 $\pm$ 5.32**	**99.70 $\pm$ 0.18**	**91.79 $\pm$ 4.55**	**98.17 $\pm$ 0.27**	39.12 $\pm$ 15.68
not-enzymes	0.5	82.68 $\pm$ 1.50	**67.99 $\pm$ 1.90**	90.12 $\pm$ 1.36	77.71 $\pm$ 2.80	**84.76 $\pm$ 0.94**	**3.58 $\pm$ 0.54**
	1	**83.17 $\pm$ 1.23**	67.52 $\pm$ 2.49	**91.09 $\pm$ 0.71**	**79.30 $\pm$ 1.83**	84.72 $\pm$ 1.07	20.93 $\pm$ 2.89

In order to properly benchmark the proposed multikernel approach, a One-Class Classification System (hereinafter OCC or OCC_System) capable of exploiting multiple dissimilarities is used. This classification system has been initially proposed in [102] and later used for modelling complex systems such as smart grids [99, 101, 102]. The main idea in order to build a model through the One-Class Classifier is to use a clustering-evolutionary hybrid technique [99, 102]. The main assumption is

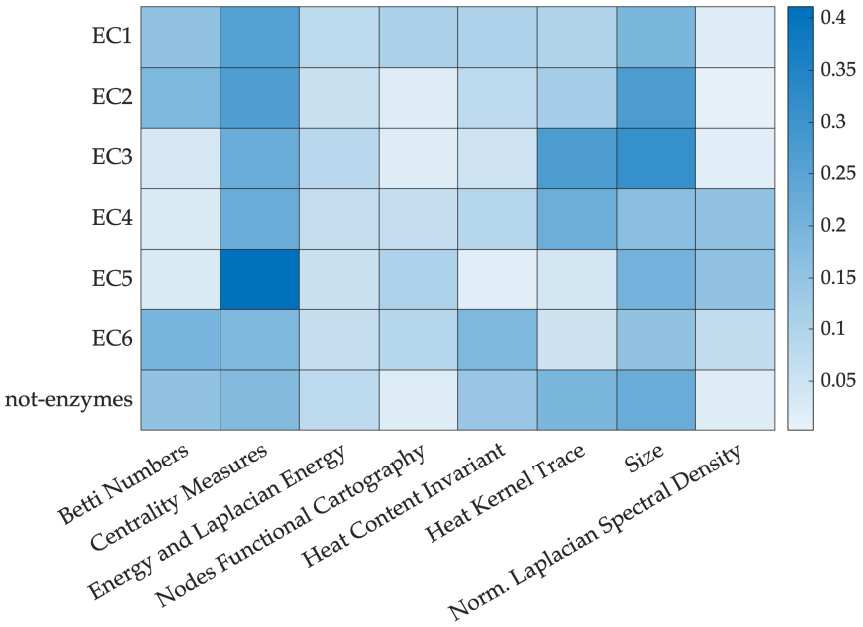


FIGURE 5.8: EC classification via multiple kernel machines, weights heatmap ($\alpha = 1$).

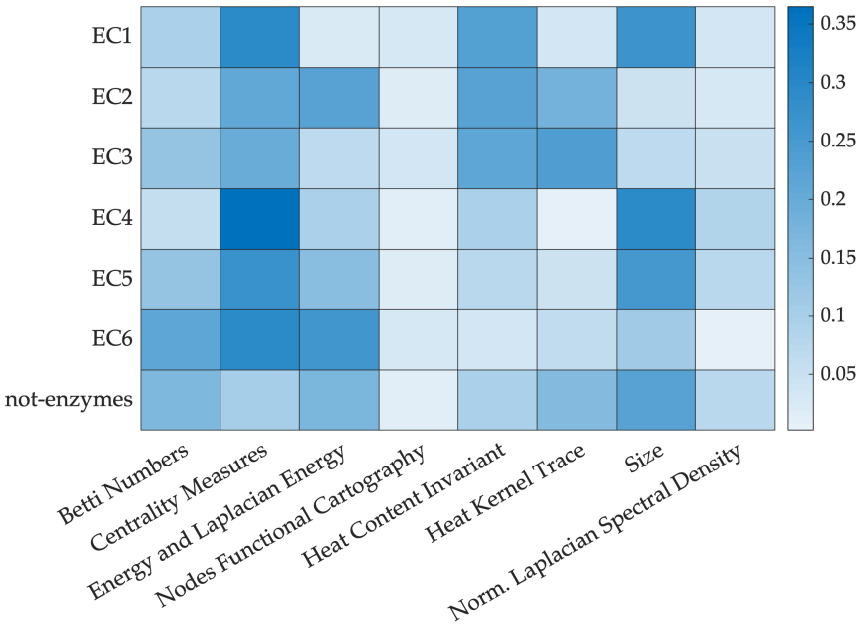


FIGURE 5.9: EC classification via multiple kernel machines, weights heatmap ($\alpha = 0.5$).

that similar protein types have similar chances of generating a specific class, reflecting the cluster model. Therefore, the core of the recognition system is a custom-based dissimilarity measure computed as a weighted Euclidean distance, that is:

$$d(\mathbf{x}_1, \mathbf{x}_2, \mathbf{W}) = \sqrt{(\mathbf{x}_1 \ominus \mathbf{x}_2)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_1 \ominus \mathbf{x}_2)} \tag{5.10}$$

where $\mathbf{x}_1, \mathbf{x}_2$ are two generic patterns and $\mathbf{W}$ is a diagonal matrix whose elements are generated through a suitable vector of weights $\mathbf{w}$. The dissimilarity measure is component-wise, therefore the $\ominus$ symbol represents a generic dissimilarity measure, tailored on each pattern subspace, that has to be specified depending on the semantic of data at hand.

In this study, patterns are represented by dissimilarity vectors extracted from each sub-dissimilarity matrix, one for each feature adopted to describe the protein. In other words, also in the OCC_System, patterns pertain to a suitable dissimilarity space.

The decision region of each cluster C_i is constructed around the medoid c_i bounded by the average radius $\delta(C_i)$ plus a threshold σ, considered together with the dissimilarity weights $\mathbf{w} = diag(\mathbf{W})$ as free parameters. Given a test pattern $\mathbf{x}$ the decision rule consists in evaluating whether it falls inside or outside the overall target decision region, by checking whether it falls inside the closest cluster. The learning procedure consists in clustering the Training Set composed by target patterns, adopting a standard genetic algorithm in charge of evolving a family of cluster-based classifiers considering the weights $\mathbf{w}$ and the thresholds of the decision regions as search space, guided by a proper objective function. The latter is evaluated on the Validation Set, taking into account a linear combination of the accuracy of the classification (that should be maximised) and the extension of the thresholds (that should be minimised). Note that in building the classification model only target patterns are used (positive), while non-target ones (negatives) are used in the cross-validation phase, hence the adopted learning paradigm is the One-Class classification one [196, 314]. Moreover, in order to outperform the well-known limitations of the initialisation of the standard k-means algorithm, the OCC_System initialises more than one instance of the clustering algorithm with random starting representatives, namely medoids since the OCC_System is capable of dealing with arbitrarily structured data.

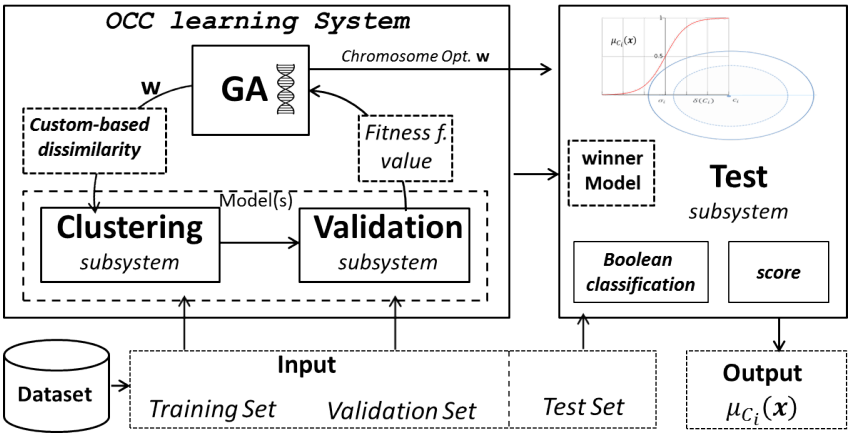


FIGURE 5.10: Schematic of the OCC_System and its learning procedure. The model provides the crisp decision as well as a score (a real number) encoding the decision reliability.

At test stage (or during validation) a voting procedure for each cluster model is performed. This technique allows building a more robust model. Figure 5.10 shows the schematic representing the core subsystems of the proposed OCC_System, such as Clustering and Genetic Algorithm. Moreover, it is shown the Test subsystem, where given a generic test pattern and given a learned model, it is possible to associate a score value (soft-decision) besides the Boolean decision. Hence, each cluster C_i is equipped with a suitable membership function, denoted in the following as $\mu_{C_i}(\cdot)$. In practice, a fuzzy set [268] is generated over C_i. The membership function allows us to quantify the uncertainty (expressed by the membership degree in $[0, 1]$) of a decision about the recognition of a test pattern. Membership values close to either 0 or 1 denote 'certain' and hence reliable decisions. When the membership degree assigned to a test pattern is close to 0.5, there is no clear distinction about

the fact that such a test pattern is really a target pattern or not (regardless of the correctness of the Boolean decision). For this purpose, we adopt a parametric sigmoid model for $\mu_{C_i}(\cdot)$, which is defined as follows:

$$\mu_{C_i}(\mathbf{x}) = \frac{1}{1 + \exp\{(d(c_i, \mathbf{x}) - b_i)/a_i\}} \tag{5.11}$$

where $a_i, b_i \geq 0$ are two parameters specific to C_i, and $d(\cdot, \cdot)$ is the dissimilarity measure from Eq. (5.10). Notably, a_i is used to control the steepness of the sigmoid (the lower the value, the faster the rate of change), and b_i is used to translate the function in the input domain. If a cluster (that models a typical protein found in the Training Set) is very compact, then it describes a very specific scenario. Therefore, no significant variations should be accepted to consider test patterns as members of this cluster. Similarly, if a cluster is characterised by a wide extent, then a tolerance might be suitable in the evaluation of the membership. Accordingly, the parameter a_i is set equal to $\delta(C_i)$. On the other hand, $b_i = \delta(C_i) + \sigma_i/2$. This allows to position the part of the sigmoid that changes faster right in-between the area of the decision region determined by the dissimilarity values falling in $[B(C_i) - \sigma_i, B(C_i)]$, where in turn $B(C_i) = \delta(C_i) + \sigma_i$ is the boundary of the decision region related to the i^{th} cluster. Finally, the soft decision function $s(\cdot)$ is defined as

$$s(\mathbf{x}) = \mu_{C^*}(\mathbf{x}) \tag{5.12}$$

where C^* is the cluster where the test (target) pattern falls.

In conclusion, the OCC_System works in two phases:

1. learning a cluster model of proteins through a suitable dataset divided into two disjoint sets, namely Training and Validation Set

2. using the learned model in order to recognise or classify unseen patterns drawn from the Test Set, assigning to each pattern a probability value.

The OCC parameters defining the model are optimised by means of a genetic algorithm guided by a suitable objective function that takes into account the classification accuracy. For the sake of comparison, the same genetic operators (selection, mutation, crossover, elitism) as per the multiple kernel system have been considered. As concerns the complexity of the model, measured as the cardinality of the partition k, a suitable value $k = 120$ has been selected.

Table 5.13 shows the comparison between the OCC_System and the multiple kernel approach (MKMD). In order to ensure a fair comparison, since the OCC_System does not perform representatives selection in the dissimilarity space, in the multiple kernel genetic code (cf. Eq. (5.9)), the weights vector $\mathbf{w}$ has been removed and all weights have been considered unitary (i.e., no representative selection). Further, the very same five Training-Validation-Test Set splits have been fed to both classifiers. Clearly, the multiple kernel approach outperforms OCC_System in terms of performances. The opposite is true as the structural complexity of the trained model is concerned[38]: indeed, whilst 120 clusters are needed in order to build the OCC model, the training phase for the multiple kernel approach returned an average of 1300 support vectors (≈52% of the training data) for EC1, 1881 support vectors (≈76%) for

[38] In brief, the structural complexity of a clustering-based classifier is strongly related to the number of clusters since, in order to classify a new pattern, the pairwise distances with respect to the centroids/medoids have to be computed. Conversely, for SVMs, one shall evaluate as many dot products as the are support vectors [113, 250].

EC2, 1745 support vectors (≈70%) for EC3, 1213 support vectors (≈49%) for EC4, 767 support vectors (≈31%) for EC5, 864 support vectors (≈35%) for EC6 and 1945 support vectors (≈78%) for not-enzymatic proteins.

TABLE 5.13: EC classification: comparison between One-Class Classifier (OCC) and the multiple kernel approach (MKMD), performances on the Test Set (avg ± std). In bold, the best between the two classifiers.

Class	Classifier	ACC	SNS	SPC	PPV	NPV
EC1	MKMD	**95.43 ± 0.27**	**67.26 ± 2.48**	**98.88 ± 0.21**	**88.02 ± 1.80**	**96.11 ± 0.28**
	OCC	91.78 ± 0.68	35.26 ± 9.93	98.7 ± 0.51	77.32 ± 3.37	92.59 ± 1.02
EC2	MKMD	**91.48 ± 0.38**	**66.22 ± 1.19**	**97.99 ± 0.30**	**89.48 ± 1.45**	**91.84 ± 0.27**
	OCC	83.49 ± 0.81	44.80 ± 5.47	93.46 ± 1.07	63.94 ± 2.68	86.80 ± 1.03
EC3	MKMD	**90.06 ± 0.38**	**57.48 ± 1.55**	**97.48 ± 0.27**	**83.89 ± 1.50**	**90.96 ± 0.30**
	OCC	83.44 ± 4.28	48.78 ± 7.69	91.34 ± 6.92	61.61 ± 15.6	88.72 ± 0.82
EC4	MKMD	**96.50 ± 0.09**	53.41 ± 0.55	**99.55 ± 0.09**	**89.42 ± 1.95**	**96.79 ± 0.04**
	OCC	68.43 ± 0.35	**77.68 ± 19.32**	44.37 ± 22.28	68.43 ± 6.64	61.09 ± 7.46
EC5	MKMD	**97.59 ± 0.11**	**43.56 ± 1.99**	**99.63 ± 0.11**	**81.92 ± 4.83**	**97.91 ± 0.07**
	OCC	85.41 ± 5.60	36.67 ± 9.16	87.25 ± 6.14	10.37 ± 1.82	97.35 ± 0.21
EC6	MKMD	**98.53 ± 0.22**	**76.72 ± 2.43**	**99.66 ± 0.19**	**92.26 ± 4.12**	**98.81 ± 0.12**
	OCC	96.88 ± 2.45	60.09 ± 18.10	98.63 ± 2.10	75.79 ± 31.45	98.11 ± 0.56
not-enzymes	MKMD	**82.11 ± 0.35**	68.12 ± 0.86	**89.58 ± 0.54**	**77.72 ± 0.84**	**84.05 ± 0.34**
	OCC	68.43 ± 0.35	**77.68 ± 19.32**	44.37 ± 22.28	68.43 ± 6.64	61.09 ± 7.46

5.2.6 Final Remarks

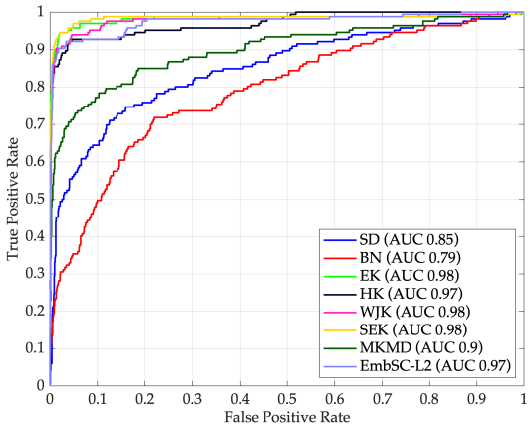
In this Section, five (hyper)graph-based pattern recognition techniques have been applied to real-world proteomic data for assessing whether it is possible to predict the EC number starting from the topological structure of a folded protein.

In order to ease the comparison between the five techniques, in Figure 5.10 the Receiver Operating Characteristic (ROC) curves [137] are shown in a class-wise fashion: each plot shows the ROC curves for a given class by considering all of the five techniques (the best of the five runs). In the legend[39], the Area Under the Curve (AUC) is also shown.

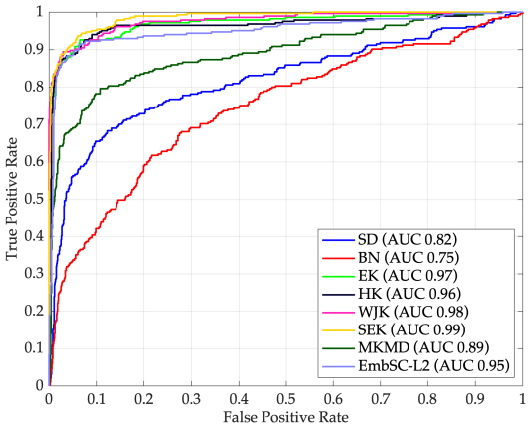
As classification performances are concerned, the three embedding-based techniques (multiple kernels, hypergraph kernels and embedding over simplicial complexes) greatly outperform the other two feature generation-based competitors (spectral density and Betti numbers), especially for EC5, where SD and BN have AUC approaching 0.5 (random chance).

All techniques other than hypergraph kernels rely on parametric dissimilarity measures whose weights have been tuned by means of a genetic algorithm. As the two feature generation based techniques are concerned, Figure 5.5 and Figure 5.7 show the corresponding weights for each feature: however, the knowledge discovery phase is rather poor in these cases (e.g., the assessment of which part of the spectral density is considered important is more 'out of curiosity' than for the sake of discoverable knowledge). The same is not true for the multiple kernel and the embedding over simplicial complexes experiments.

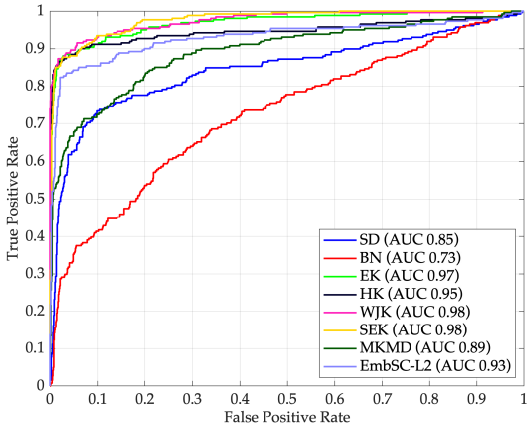
[39] Abbreviations: Spectral Density (SD), Betti Numbers (BN), Edit Kernel (EK), Histogram Kernel (HK), Weighted Jaccard Kernel (WJK), Stratified Edit Kernel (SEK), Multiple Kernels over Multiple Dissimilarities at $\alpha = 1$ (MKMD), Embedding over Simplicial Complexes with ν-SVMs at $\alpha = 1$ (EmbSC-L2).



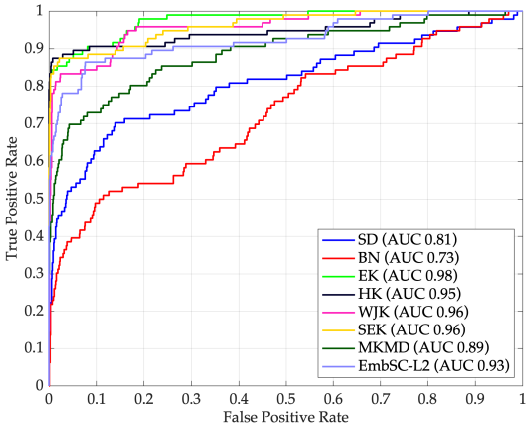
(A) EC1



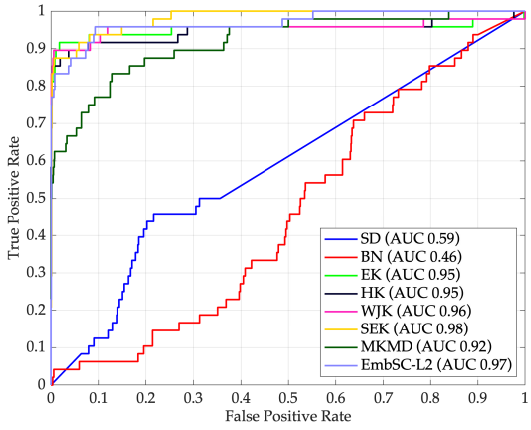
(B) EC2



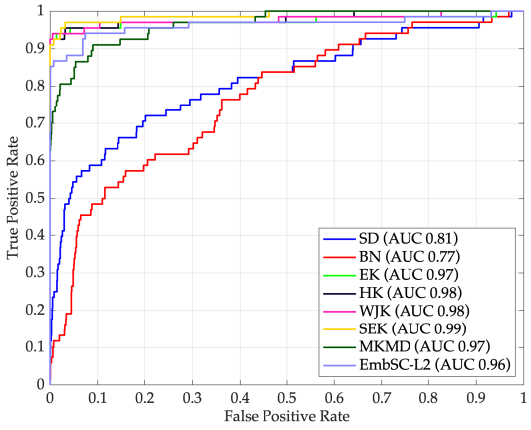
(C) EC3



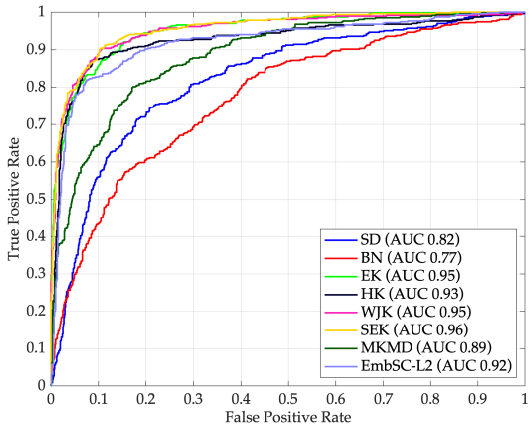
(D) EC4



(E) EC5



(F) EC6



(G) not-enzymes

FIGURE 5.10: ROC curves for EC number classification.

In the first case, one can easily determine by analysing the kernel weights (see Figures 5.8–5.9) which representation is the most relevant for the problem at hand. Furthermore, by analysing the weights vector, one can determine which patterns have been considered as pivotal for the dissimilarity space. As thoroughly discussed in the reference paper [261], the protein size and the centrality measures seems to be the most important characteristics: this is perfectly in line with current biological knowledge if we consider that enzymes have a more pronounced allosteric effect with respect to non-enzymatic proteins. This is a consequence of the need to modulate chemical kinetics according to microenvironment conditions (allostery is the modulating effect of a modification happening in a site different from catalytic site on the efficiency of the reaction [159]). Allostery implies an efficient transport of the signal along protein structure and it was discovered to be efficiently interpreted in terms of PCN descriptors, especially centrality measures [285]. Further, by analysing the proteins marked as pivotal for the dissimilarity spaces, some interesting conclusions can be derived as well, despite the classification task is indeed very hard (the ability to catalyse a specific class of chemical reaction has a small impact on the overall global shape). Regardless of the target class, the pivotal proteins come from all EC classes and not only from the target class: this is perfectly in line with both biological knowledge (absence of a clear form-function relation, hence they can be considered as an 'emergent property' of the discrimination task) and pattern recognition knowledge (the pivotal patterns in the dissimilarity space must well characterise the decision boundary between the two classes). The presence of molecules of different classes crucial for a specific category modelling and thus the image in light of a peculiar strategy adopted by the system is analogue to the use of 'paired samples' in statistical investigation [33, 110]. When in presence of only minor details discriminating statistical units pertaining to different categories, the only possibility to discriminate is to adopt a paired samples strategy in which elements of a category is paired with a very similar example of another category so to rely on their differences (on a sample-by-sample basis) instead of looking for a general 'class-specific' properties. The three basic structural patterns found in the pivotal proteins (spherical shape, symmetrical shape and elongated shape), despite having different frequencies in the EC-classes[40], are present in all analysed classes in order to ensure the sample-by-sample analysis discussed above. This peculiar situation is in line with current biochemical knowledge (minimal effect exerted by

[40]For example, the elongated shape is way more common in non-enzymatic proteins.

catalysed reaction on global structure) and it is a relevant proof-of-concept of both the reliability of the classification solution and of the power of the proposed multiple kernels/multiple dissimilarities approach. Appendix A (Section A.1) enlists the most relevant proteins marked as pivotal for the dissimilarity space.

Similar knowledge discovery can be carried out on the simplicial complexes embedding experiment, thoroughly discussed in the reference paper [252], in which one may ask why the selected simplices have been considered as relevant for the classification problem at hand. In order to extract a biochemically relevant explanation from the results of the pattern recognition procedure, for each extracted granule (simplex), namely a small peptide located into the protein structure, the main chemico-physical parameters at the amino-acid residue level (volume, polarity, hydrophilicity) according to the results presented in [30] have been computed. Each information granule (simplex) has been mapped with 6 real values indicating the average and standard deviation of polarity, volume and hydrophilicity evaluated amongst the amino-acids forming the simplex. The chemico-physical properties of each information granule have been correlated with a score ranging from 1 to 5, namely the number of times said granule has been selected across the five test runs: the higher the score, the higher the confidence about its discrimination importance for the classification problem. Seven statistical models (one for each of the problem-related classes) based on linear discrimination analysis [140]: all of these models show statistical significance, mainly thanks to the large number of samples (more than 12000 simplices). Table 5.14 summarises their main characteristics. Alongside the statistical significance, it is interesting to note that *all* of the seven models have $R^2 \approx 0.02$, meaning that they explain 2% of the overall variance.

TABLE 5.14: Variance explained and statistical significance for the seven models.

	EC1	**EC2**	**EC3**	**EC4**	**EC5**	**EC6**	**not enzymes**
R^2	0.0250	0.0239	0.0212	0.0199	0.0239	0.0170	0.0250
p	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

In this experiment (and the same will also hold for the solubility prediction experiment, see later Section 5.3.5) hydrophilicity has been shown to be the most important predictor and completely superimposable results are obtained for average polarity that is strictly related to hydrophilicity. Table 5.15 shows the main characteristics of the seven models as hydrophilicity is concerned and Table 5.16 is its counterpart as regards polarity: they both reports the t-statistics and the relative p-value of the null hypothesis of no contribution of hydrophilicity (polarity) of the multiple linear regression having score for different classes as dependent variable and different chemico-physical indexes relative to the simplices as regressors. As evident, especially hydrophilicity, enters with a significant contribution in all models, stemming as the most important predictor (i.e., the estimated coefficient for average hydrophilicity is approximately one order of magnitude higher with respect to other coefficients). Another interesting aspect is that all models show a negative coefficient for average hydrophilicity and positive sign for its standard deviation.

TABLE 5.15: Hydrophilicity contribution to score for different classes.

	Hydrophilicity (avg)			Hydrophilicity (std)		
Class	t-value	p	Coefficient	t-value	p	Coefficient
1	11.55	< 0.0001	-4.17734	0.92	0.3563	0.24438
2	10.52	< 0.0001	-3.73211	0.0647	1.85	0.47999
3	10.61	< 0.0001	-3.38981	0.0651	1.84	0.43182
4	11.08	< 0.0001	-2.98596	2.11	0.0352	0.41574
5	12.13	< 0.0001	-2.43624	2.49	0.0127	0.36671
6	10.73	< 0.0001	-2.65512	2.57	0.01	0.46672
7	11.55	< 0.0001	-4.17734	0.92	0.3563	0.24438

TABLE 5.16: Polarity contribution to score for different classes.

	Polarity (avg)			Polarity (std)		
Class	t-value	p	Coefficient	t-value	p	Coefficient
1	11.27	< 0.0001	1.51515	1.77	0.0762	-0.17376
2	10.26	< 0.0001	1.35280	2.52	0.0118	-0.24206
3	10.43	< 0.0001	1.23898	2.62	0.0089	-0.22655
4	10.83	< 0.0001	1.08515	2.72	0.0066	-0.19836
5	11.84	< 0.0001	0.88388	3.16	0.0016	-0.17190
6	10.52	< 0.0001	0.96768	3.14	0.0017	-0.21080
7	11.27	< 0.0001	1.51515	1.77	0.0762	-0.17376

5.3 PCN Experiments: Solubility Degree

An interesting aspect of the pattern recognition techniques discussed in Chapter 4 is that almost all of them are suitable for solving also function approximation and clustering problems, alongside classification problems as in Section 5.2. The only exception is the INDVAL-based technique: indeed, in order to evaluate the INDVAL score, one needs a finite number of problem-related classes (see Eqs. (4.16)–(4.18)), a peculiarity that only classification problems exhibit (cf. Section 3.1). In order to see the proposed pattern recognition techniques in action for function approximation problems, aim of this Section is to investigate the relationship between the PCN and its solubility degree.

5.3.1 Solubility Prediction via Spectral Density

From the baseline PCN-SOL dataset (4781 patterns), 26 patterns have been removed because they had at least one isolated node and the evaluation of $\mathbf{D}^{-1/2}$ is impossible (as per Section 5.2.1), leading to a total number of 4755 proteins. In order to perform a stratified splitting between Training, Validation and Test Set, the ground-truth values (solubility degrees) have been distributed into 10 uniformly spaced bins in $[0, 1]$ and the ID of the bin serves as a '(fake) ground-truth label'. Due to the finite and categorical nature of this labelling, the stratified splitting has been possible, keeping the same proportions of 50% (Training Set), 25% (Validation Set) and 25% (Test Set). Once the three sets are returned, the original solubility degrees have been

considered as the proper ground-truth values, whereas the ID of the bins have been discarded.

Methodologically speaking, the model synthesis does not change with respect to the EC classification experiment from Section 5.2.1: for each PCN within each split, $m = 100$ samples are drawn from the normalised Laplacian spectral density.

Since this is a function approximation problem rather than a classification problem, the core model is a radial basis function-kernelised ν-Support Vector Regression [336] (ν-SVR) and the following two performance indices take place in lieu of accuracy, sensitivity, informedness and other indices from Section 5.2:

Mean Squared Error: $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

Squared Correlation Coefficient: $R^2 = \frac{\left(N\sum_{i=1}^{N} y_i\hat{y}_i - \sum_{i=1}^{N}\hat{y}_i\sum_{i=1}^{N} y_i\right)^2}{\left(N\sum_{i=1}^{N}\hat{y}_i^2 - \left(\sum_{i=1}^{N}\hat{y}_i\right)^2\right)\cdot\left(N\sum_{i=1}^{N} y_i^2 - \left(\sum_{i=1}^{N} y_i\right)^2\right)}$

where N is the number of samples, y and $\hat{y}$ indicate the true and predicted output value, respectively.

The genetic code for the optimisation procedure still reads as Eq. (5.3) and the genetic algorithm setup remains unchanged. Trivially, the only difference relies on the fitness function due to the unfeasibility of the informedness for non-classification problems: for this experiment, the R^2 (to be maximised) serves as the fitness function[41].

Table 5.17 shows the average and standard deviation of both MSE and R^2 on the Test Set evaluated across five Training-Validation-Test Set splits. For the sake of completeness, as in Figure 5.5, Figure 5.11 shows the resulting weights.

TABLE 5.17: Solubility prediction via spectral density, performances on the Test Set (avg $\pm$ std).

MSE	$\boldsymbol{R^2}$
0.0534 ± 0.0017	0.4439 ± 0.0196

5.3.2 Solubility Prediction via Betti Numbers

This experiment is the counterpart of the experiment described in Section 5.2.2, in which the Betti numbers are evaluated on the Vietoris-Rips Complex at different scales $\epsilon = \{4, 5, 6, 7, 8\}$. The Clique Complex will not be tested, due to its poor performances.

From the initial set of 4781 proteins in PCN-SOL, 5 proteins have been removed because of out-of-memory errors during the evaluation of the three Betti numbers at $\epsilon = 8$. The remaining 4776 proteins have been cast towards the 15-length integer-valued vector already in Eq. (5.5) which contains the first three Betti numbers for each candidate ϵ.

The resulting dataset has been split in Training, Validation and Test Set following the same binning strategy described in Section 5.3.1. The genetic algorithm orchestration does not change with respect to the previous experiment (Section 5.3.1).

[41] The choice behind the R^2 over the MSE follows: the R^2 is bounded in $[0, 1]$ and the closer to 1, the better. The same is generally not true for the MSE, for which "the lower, the better" holds. Hence, the R^2 allows for a more informed evaluation of the goodness of fit.

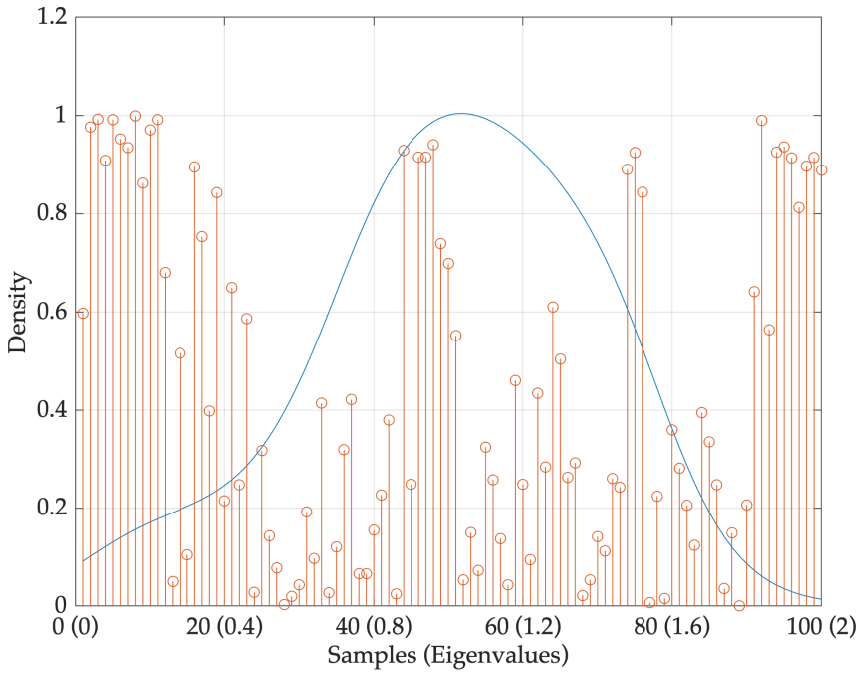


FIGURE 5.11: Solubility prediction via spectral density, weights vector. Along with the weights values (orange stems), the average spectral density amongst the entire dataset is also shown.

Table 5.18 shows the average and standard deviation of both MSE and R^2 on the Test Set evaluated across five Training-Validation-Test Set splits. For the sake of completeness, as in Figure 5.7, Figure 5.12 shows the resulting weights.

TABLE 5.18: Solubility prediction via Betti numbers, performances on the Test Set (avg $\pm$ std).

MSE	R^2
0.0613 ± 0.0019	0.3706 ± 0.0164

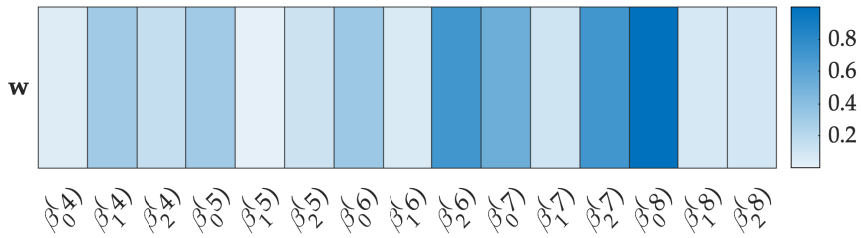


FIGURE 5.12: Solubility prediction via Betti numbers, weights vector.

5.3.3 Solubility Prediction via Embedding over Simplicial Complexes

This experiment is the counterpart of the experiment described in Section 5.2.3, in which each PCN has been described as the histogram of simplices composing the Clique Complex.

The entire 4781 proteins in PCN-SOL have been considered and properly split into Training, Validation and Test Set according to the already-mentioned binning strategy. For each PCN, the Clique Complex is evaluated and vertices are represented according to the amino-acid type. The set of unique simplices from the union of Training Set and Validation Set forms the alphabet $\mathcal{A}$ on the top of which the symbolic histograms are individually evaluated for each simplicial complex. Whilst for the EC number classification the alphabet counts approximately 12000 symbols, in this case it counts approximately 11000 symbols.

The genetic algorithm wrapping the model synthesis procedure relies on a genetic code that reads as in Eq. (5.6) and the setup remains unaltered with respect to the embedding over simplicial complexes for EC number classification (Section 5.2.3). The fitness function, akin to Eq. (5.7), (still to be minimised) sees a convex linear combination between $(1 - R^2)$ and the sparsity.

Table 5.19 shows the performances (in terms of average and standard deviation) on five Training-Validation-Test splits. Both $\alpha = 0.5$ and $\alpha = 1$ in the fitness function have been used for comparison. The sparsity (i.e., the percentage of selected symbols) is shown as well.

TABLE 5.19: Solubility prediction via embedding over simplicial complexes, performances on the Test Set (avg $\pm$ std).

	MSE	R^2	**Sparsity**
$\alpha = 0.5$	$0.0298 \pm 9.8396 \cdot 10^{-4}$	0.6919 ± 0.0101	6.8272 ± 1.8437
$\alpha = 1$	0.0251 ± 0.0015	0.7437 ± 0.0168	41.2517 ± 17.4360

5.3.4 Solubility Classification via Embedding over Simplicial Complexes

In Sections 5.3.1–5.3.3, the solubility prediction problem has been formulated as a function approximation problem, where target of the learning system is to predict the solubility degree (recall, a scalar in range $[0, 1]$) as accurately as possible. However, it is possible to 'relax' this task by letting the learning system to discriminate between soluble and non-soluble proteins. In other words, this 'relaxed' problem turns into a binary classification problem where proteins having solubility degree greater than a user-defined threshold τ are considered 'soluble', whereas the remaining proteins are marked as 'non-soluble' [252].

For the sake of shorthand, only the embedding over simplicial complexes (one of the most promising tecniques, according to Section 5.2) is considered, with ℓ_1-SVM acting as the core classification model due to efficiency (very fast training times, as stressed towards the end of Section 5.2.3) and effectiveness (comparable performances with non-linear methods, as can be seen by matching Tables 5.6 and 5.7).

For a thorough investigation, the solubility threshold τ has been varied in range $[0.1, 0.9]$ with step size 0.1, for a total of 9 candidate values. For each candidate τ, the entire PCN-SOL dataset has been relabelled (soluble vs. non-soluble proteins) and split in Training-Validation-Test Set according to the usual stratified sampling. After performing the embedding, the classification is still orchestrated via genetic optimisation where the genetic code has the form (alike Eq. (5.8))

$$\begin{bmatrix} C & c_- & c_+ \end{bmatrix} \tag{5.13}$$

where c_- and c_+ are the two additional weights in order to adjust the regularisation term C for negative and positive misclassifications, respectively. The fitness function is given by Eq. (5.7). Both $\alpha = 0.5$ and $\alpha = 1$ have been considered for testing.

Figures 5.13 and 5.14 show the classification performances and the percentage of selected symbols (sparsity) as function of τ. As usual, for each τ five Training-Validation-Test splits have been performed and the average values are shown.

By matching the top panels in Figures 5.13 and 5.14, the best threshold values are in range $\tau \in [0.5, 0.7]$ for $\alpha = 1$ and $\tau \in [0.5, 0.6]$ for $\alpha = 0.5$. In the latter case, as $\tau \to 0.7$, precision starts deteriorating. Indeed, for very low threshold values ($\tau \to 0.1$) there will be a lot of 'soluble' proteins with respect to 'non-soluble' ones[42]: trivially, this reflects in very high positive-related performance indices such as precision and sensitivity (circa 100%) and rather low negative-related performance indices such as negative predictive value and specificity (circa 80-90%). The opposite is true for very high thresholds ($\tau \to 0.9$). In the aforementioned ranges, all performance indices are rather balanced: in Figure 5.13, for $\tau \in [0.5, 0.7]$, all performance indices are in range 89-94%; in Figure 5.14, for $\tau \in [0.5, 0.6]$, all performance indices are in range 89-92%. This (minor) shift in performances is counterbalanced by the number of selected symbols: for $\alpha = 1$ approximately 20% of the alphabet symbols have been selected, whereas for $\alpha = 0.5$ the percentage of selected symbols is always below 5%. Interestingly, in Figure 5.14, the range $\tau \in [0.5, 0.7]$ is also featured by the largest alphabet: a slightly more complex embedding space is needed for maximising the overall performances.

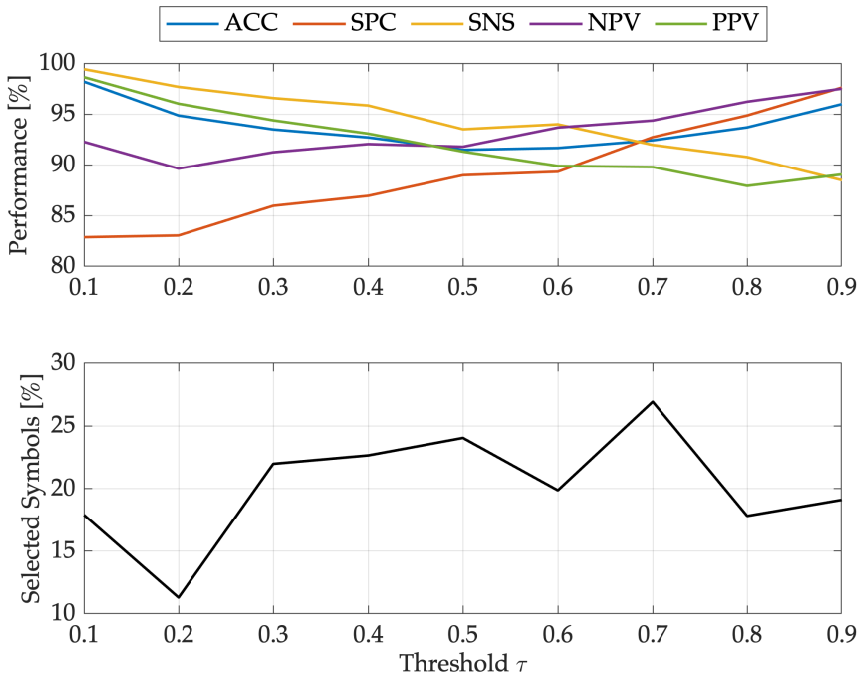


FIGURE 5.13: Solubility classification via embedding over simplicial complexes, average performances as function of τ ($\alpha = 1$).

[42] In other words, many positive instances with respect to negative ones.

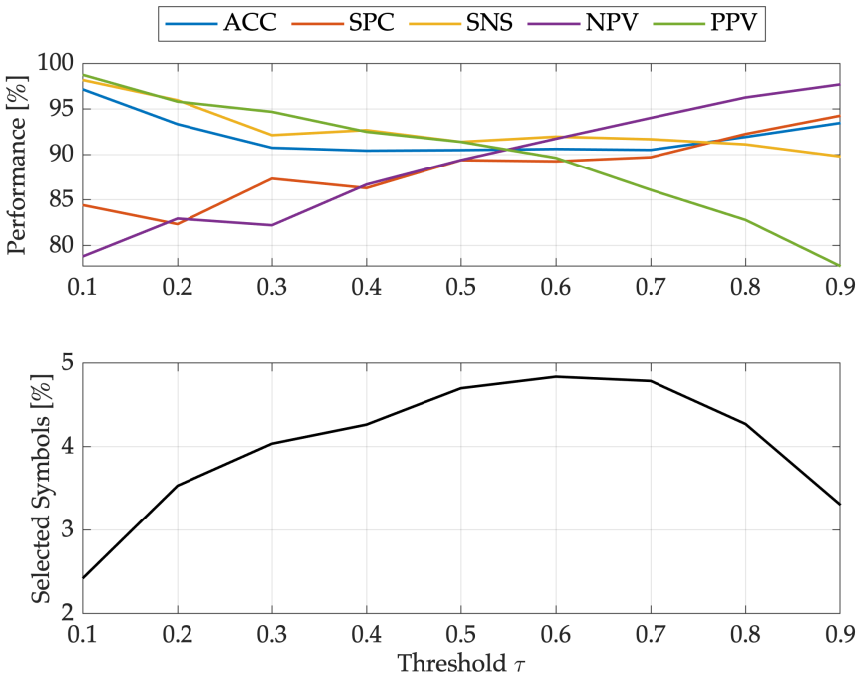


FIGURE 5.14: Solubility classification via embedding over simplicial complexes, average performances as function of τ ($\alpha = 0.5$).

5.3.5 Final Remarks

In this Section, three out of the six proposed pattern recognition techniques from Chapter 4 have been tested in a function approximation scenario, conversely to the classification scenario in Section 5.2. This has been possible because in (almost) all of the cases, the feature space is independent of the nature of the problem at hand with the only exception being, as already introduced at the beginning of Section 5.3, the INDVAL-based strategy that demands finite and categorical class labels.

Methodologically speaking, moving from the classification scenario towards the function approximation scenario demands two major workflow modifications:

1. changing the pattern recognition model (e.g., from ν-SVM to ν-SVR)
2. changing the fitness function (e.g., from informedness to R^2).

For the sake of argument, it is worth saying that despite unsupervised problems have not been explicitly addressed, such pattern recognition procedures can also be employed for clustering problems by applying the very same two modifications above (i.e., selecting a suitable clustering algorithm along with a suitable performance index).

Leaving aside the INDVAL-based strategy, only three of the remaining five techniques have been tested, namely the two feature engineering approaches (spectral density –Section 5.3.1– and Betti numbers –Section 5.3.2) and the embedding over simplicial complexes (Section 5.3.3): for the sake of shorthand, only the two worst-performing techniques (spectral density and Betti numbers) and one of the best-performing techniques (embedding over simplicial complexes) according to the EC classification experiments from Section 5.2 have been considered. Despite the different nature of the problem (function approximation vs. classification), by matching Tables 5.17, 5.18 and 5.19, the embedding over simplicial complexes still emerges as the most performing technique.

Following the same rationale, in Section 5.3.4, only the embedding over simplicial complexes has been tested for the classification between soluble and non-soluble proteins. Furthermore, it is worth recalling that the embedding over simplicial complexes is one of the few methods that allows a suitable knowledge discovery phase. Following the same statistical assessment as per the EC number classification experiment (Section 5.2.6, the set of selected simplices at suitable τ values have been correlated with the average and standard deviation of the different chemico-physical parameters at the amino-acid residue level. In Section 5.2.6, only polarity and hydrophilicity have been deemed of interest for the analysis at hand because volume has been shown to be not statistically significant (p-value approx. 0.11): this is perfectly coherent if we consider that the volume of a simplex (usually less than 5 residues) is very unlikely to endow biological meaning in terms of the overall protein solubility. On the other hand, the standard deviation volume has been shown to be statistically significant (p-value < 0.0001): this interesting result remarks that simplices composed by 'similar amino-acids' (small standard deviation) show better solubility. Nonetheless, it is important to note that, for a given chemico-physical property (e.g., volume in this case) the standard deviation and the average value shall be treated independently and do not show any correlation. This latter aspect of average and standard deviation carrying different information has also been confirmed by analysing the two other properties (polarity and hydrophilicity).

TABLE 5.20: Pearson correlation coefficients between polarity and hydrophilicity.

	Polarity (avg)	**Hydrophilicity (avg)**	**Polarity (std)**	**Hydrophilicity (std)**
Polarity (avg)	1	0.99818	-0.01869	-0.06879
Hydrophilicity (avg)	0.99818	1	-0.03705	-0.08582
Polarity (std)	-0.01869	-0.03705	1	0.99397
Hydrophilicity (std)	-0.06879	-0.08582	0.99397	1

Polarity and hydrophilicity not only show statistical significance (all p-values are less than 0.0001), but also show a strong correlation (> 0.99) in terms of both mean values and standard deviations, as shown in Table 5.20, yet mean values and standard deviations are not correlated with each other (as per the volume case). This perfectly fits with current biochemical knowledge and, specifically, this is consistent with the well-known importance of 'hydrophobic interaction' in protein folding (residues with hydrophobicity/hydrophilicity values tend to aggregate [284]).

In conclusion, beside the confirmation of the pivotal role of residue hydrophilic character in determining the protein structure, is well known [80] that shifting from single residue to entire protein level, new organisation principles arise and 'context-dependent' features largely overcome single residue level properties. The 2% of variance explained observed in the EC case study is the percentage which can be imputed to the plain chemico-physical properties of individual simplices and one might ask whether the same analyses can be carried by considering 'groups of simplices' instead of individual simplices and scoring their relevance for the problem at hand: this paves the way to new granulation-based studies which should take into

account also these aspects. All in all, the observed results confirm the actual biochemical theory, thus give a 'lateral validation' to the pattern recognition procedure, while at the same time push biochemists to look for non-local chemico-physical properties for getting rid of protein folding and structure-function relation.

5.4 Metabolic Pathways Experiments

5.4.1 A Preliminary Investigation on the Gut Microbiota

In Section 1.3 the network formalism for metabolic pathways has been introduced, where edges mark the existence of chemical reactions transforming a node (metabolite) i into another metabolite j. Regardless of the orientation of the edges (i.e., whether transformations are considered reversible or not), this representation has been demonstrated useful in many different scenarios [299, 337]. As such, each metabolic network can be represented by a binary adjacency matrix $\mathbf{A}$ having as many rows and columns as the are nodes which in position $\mathbf{A}_{i,j}$ sees a unit value if an edge exists between the i^{th} and the j^{th} node. Following [360], in order to compare metabolic networks with a different number of metabolites, all graphs are projected into $n \times n$ adjacency matrices, with n being the total number of metabolites present at least once in the analysed dataset: if a given metabolite k is not present in the metabolic network of a particular organism, the k^{th} row and the k^{th} column of its adjacency matrix are set to 0. The maximal coverage set of nodes trick allows a straightforward comparison between different adjacency matrices by means of the Hamming distance: the distance between two matrices (networks) is given by the number of discrepancies (i.e., different values at corresponding positions).

In order to demonstrate of the ability of the adopted representation and metric to reconstruct phylogenesis at different magnification scales, the four datasets from Section 5.1.3 have been analysed by means of the pairwise F-statistic by considering the between- and within- classes metabolic distances. Specifically, if l_i and l_j are the i^{th} and j^{th} classes for a given classification problem, their F-statistic reads as

$$F(l_i, l_j) = \frac{\overline{d}(l_i, l_j)}{0.5 \cdot (\overline{d}(l_i, l_i) + \overline{d}(l_j, l_j))} \tag{5.14}$$

where $\overline{d}(a, b)$ indicates the average (Hamming) distance amongst patterns belonging to classes a and b. Figures 5.15–5.18 show the consistency between metabolic network wiring (phenotypic) classification to reproduce the phylogenetic (genotypic) classification of organisms at different scales of definition. The F-statistic shows a clear-cut correlation between standard biological classification and metabolic data, thus confirming that both the representation and the adopted metric hold from very rough (between kingdoms) to very detailed (inside bacteria) problems.

Given these encouraging results, the next study aims at addressing the feasibility and the biological relevance of the gut microbiota in terms of the difference in metabolic network wiring of different micro-organisms. The analysis of microbiota from both methodological (wide use of biodiversity indexes [361], multidimensional statistics approaches [266]) and theoretical (contemporary presence of different species having mutualistic and/or competitive interactions, relevance of (micro) environmental conditions) viewpoints can be intended as a specialised branch of ecology [108, 136]. In this respect, when evaluating the changes induced by different

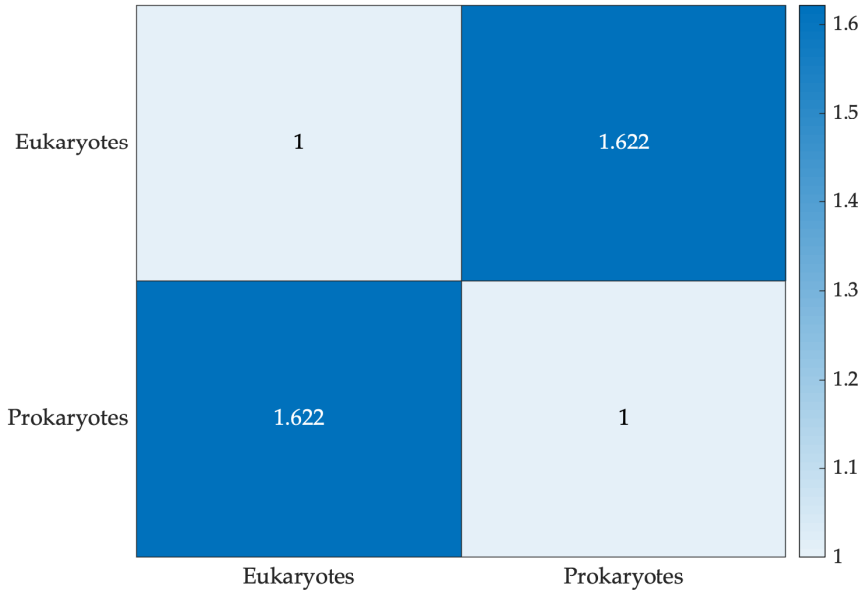


FIGURE 5.15: *F*-statistic for metabolic pathways data (Problem 1).

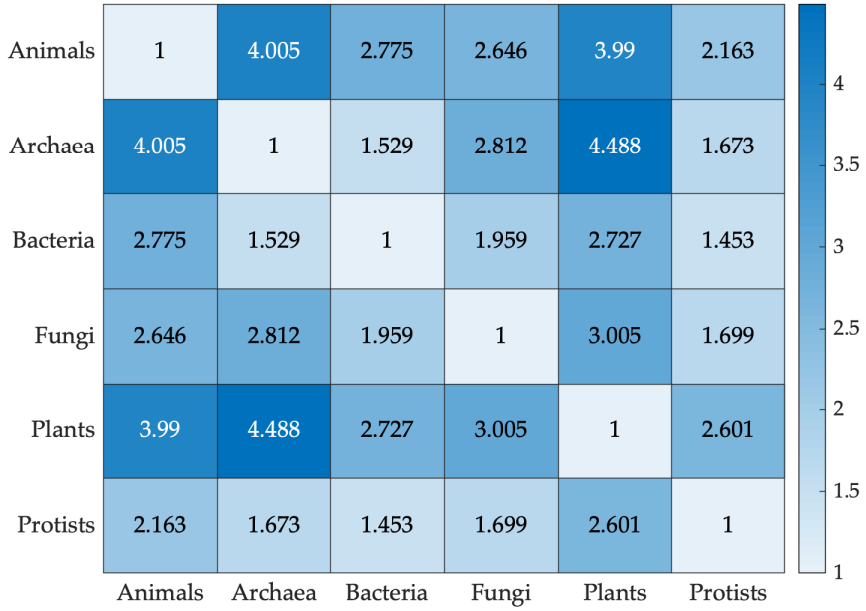


FIGURE 5.16: *F*-statistic for metabolic pathways data (Problem 2).

stimuli (e.g., disease, drugs, dietary habits...) on microbiota composition it is important to go further the simple registration of microbiota profiles in terms of phylogenetic information (e.g., changes in relative frequency of microbial genera and/or species) shifting to an appreciation of functional changes of the microbiota. Analogously to classical ecology, where species occupy 'niches' correspondent to peculiar roles in the ecological interaction web, one can hypothesise [136] that to each bacterial species corresponds a given 'niche' in the mucosa environment. That is to say that a 'healthy' microbiota is expected to have a balanced occupation of the relevant 'niches'. Exactly as in classical ecology, the phylogenetic characterisation it is not independent of the functional role played by the organisms, but it is not coincident with it. This is why the representation of microbiome in terms of patterns of relative frequencies of functional classes could represent a more 'phenotypic-oriented'

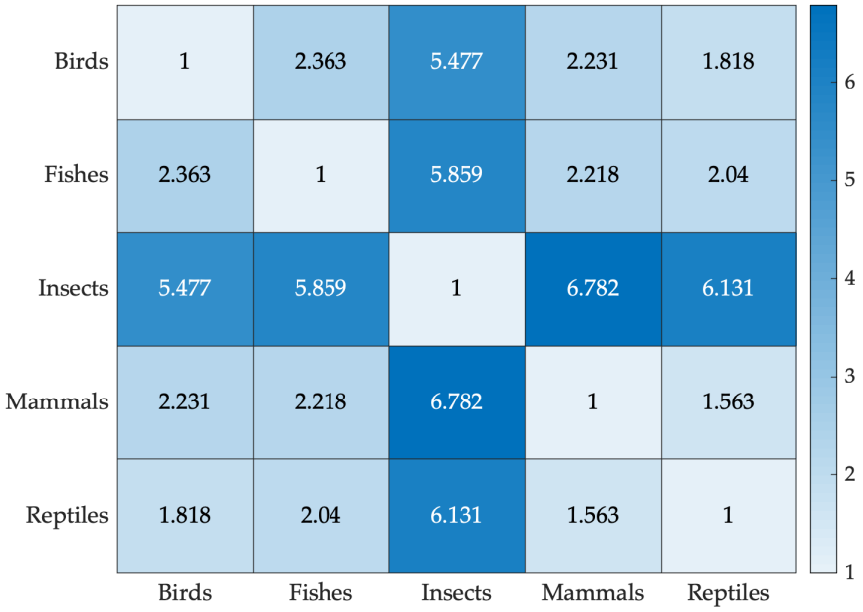


FIGURE 5.17: *F*-statistic for metabolic pathways data (Problem 3).

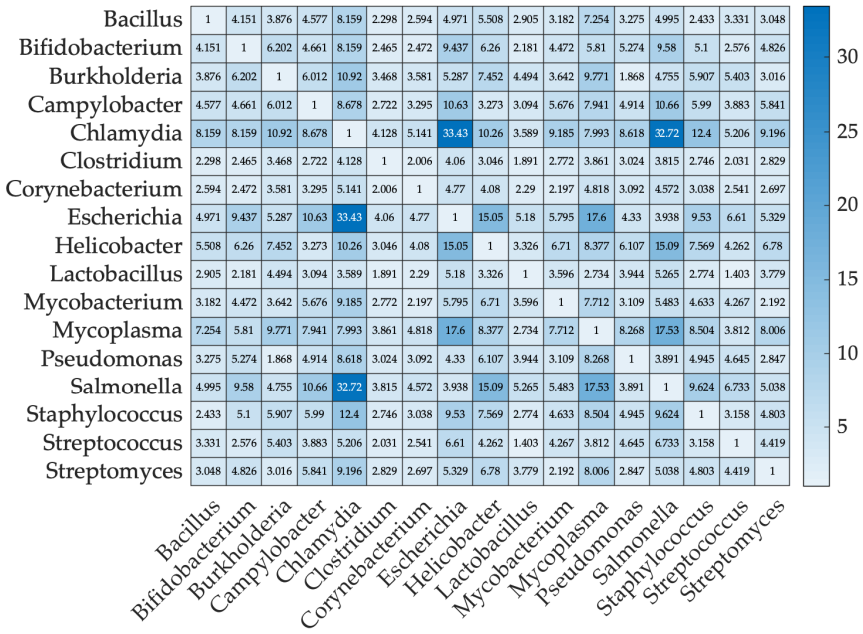


FIGURE 5.18: *F*-statistic for metabolic pathways data (Problem 4).

(and thus potentially more biologically relevant) coding of microbiota with respect to the phylogenesis oriented one. In the case of microbial species in the gut (the specific environment under analysis), the best 'proxy' of their ecological role is their global metabolic network that mirrors the specific 'niche' occupied by the organism in terms of chemical products (output) and reactants (input) as well as of their intermediate steps. This is why different microbial species are correlated by the mutual similarities of their metabolic networks wiring: the aim of this study is indeed to address whether distinct network clusters are significantly correlated with the phylogenetic classification. If successful, this is a proof-of-concept of the existence of relevant ecological niches in the microbial ecology of the gut.

From the initial dump of 5299 organisms (Section 5.1.3), 1027 have been retained

as they are known to populate the human gastrointestinal tract[43]. In order to study whether the gastrointestinal tract tends to form well-formed clusters, a k-medoids algorithm has been used. Indeed, k-medoids, while retaining the properties of partitional clustering algorithms (i.e., a given pattern belongs to one and only one cluster), does not depend to any algebraic structures (e.g., the mean in case of the well-known k-means) and can therefore be equipped with a custom dissimilarity measure [255–257]. Consequently, the binary adjacency matrices for the properly selected 1027 organisms have been considered, along with a k-medoids equipped with the Hamming distance. As in [107, 256, 257, 301], the k-medoids exploits the very same Voronoi iterations at the basis of k-means:

1. k patterns are randomly[44] drawn from the dataset as initial medoids
2. each pattern is assigned to the closest cluster (medoid)
3. all medoids are updated: the medoid (also MinSoD) is the element which minimises the sum of pairwise distances amongst the patterns belonging to the cluster
4. steps 2–3 are repeated until a maximum number of iterations is reached or a given stopping criterion is triggered (e.g., medoids stop changing).

The optimal number of clusters has been determined thanks to the 'elbow plot' heuristic [357]: a set of candidates $k = \{1, 2, \ldots, 20\}$ has been exhaustively tested and the optimal $k^\star$ is selected in proximity of an 'elbow' in the within-clusters sum-of-distances plot, drawn as function of k. The within-clusters sum-of-distances (WCSoD) is formally defined as

$$WCSoD = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{m}^{(i)})^2 \tag{5.15}$$

where C_i indicates the i^{th} cluster and $\mathbf{m}^{(i)}$ is its medoid, whereas $d(\cdot, \cdot)$ indicates the dissimilarity measure adopted (i.e., the Hamming distance, in this case).
Basically, the rationale behind the elbow plot heuristic suggests that is pointless to increase the number of clusters (i.e., the model complexity) if adding more clusters does not lead to a better modelling of the data. As usual in k-clustering, since the clustering results are sensitive to the initial conditions (see Step 1 for the Voronoi iterations), for each candidate k, 10 runs of the algorithm are performed with different initial medoids: the best run is selected as the one that better minimises the WCSoD as in Eq. (5.15).

Figure 5.19 shows the WCSoD (and its gradient, that sometimes helps in finding the elbow) as function of k with the elbow somehow visible at $k^\star = 7$, meaning that seven well-defined clusters have been found. For the $k^\star = 7$ clusters solution, which explains 75% of the overall variance, Figure 5.19 shows the proportion of genera across the seven clusters. It is immediate to note that the metabolic (ecological) clusters, while presenting very different profiles in terms of genera composition carry independent information with respect to the classical phylogenetic-based classification. This additional information must be ascribed to 'niche similarities' only partially recovered by phylogenetic based profiles and potentially useful to give a

[43] According to https://en.wikipedia.org/wiki/List_of_human_microbiota#Gastrointestinal_tract. Accessed 28/06/2018.

[44] A uniform random sampling has been employed for the sake of ease. However, there exist more efficient initialisation techniques, see e.g. [11] and [40], at an additional computational cost.

rational basis to the observed differences in microbiome induced (or mirrored) by changes in microbiota. Overall, this partition is suggestive of the existence of discrete metabolic roles played by different species functional to a balanced microbiome ecology or, in other words, the presence of a strongly clustered structure with few relevant clusters is a proof of the tenability of the hypothesis of discrete niches defining the microbiota of the gut.

In order to further explore the clustering solutions, one can ask whether there are some reactions that are specific to organisms belonging to a given cluster: this is where the INDVAL score plays the main role. Since chemical reactions are edges within a metabolic network, the INDVAL definition previously given in Section 4.4 can easily be restated as follows:

$$A_{i,j} = \frac{\#\text{ patterns in cluster } j \text{ having reaction } i}{\#\text{ patterns having reaction } i} \tag{5.16}$$

$$B_{i,j} = \frac{\#\text{ patterns in cluster } j \text{ having reaction } i}{\#\text{ patterns belonging to cluster } j} \tag{5.17}$$

$$I_{i,j} = A_{i,j} \cdot B_{i,j} \tag{5.18}$$

For each edge (reaction), the INDVAL score has been evaluated and in Table 5.21 the most relevant reactions[45] in each cluster, along with their INDVAL score, are shown. In all clusters other than the 6^{th}, there exist signature chemical reactions which, in most of the cases, are also 'perfect': the INDVAL score reaches 1, its maximum value, meaning that all members of the cluster have that reaction and none of the other clusters' members have that reaction.

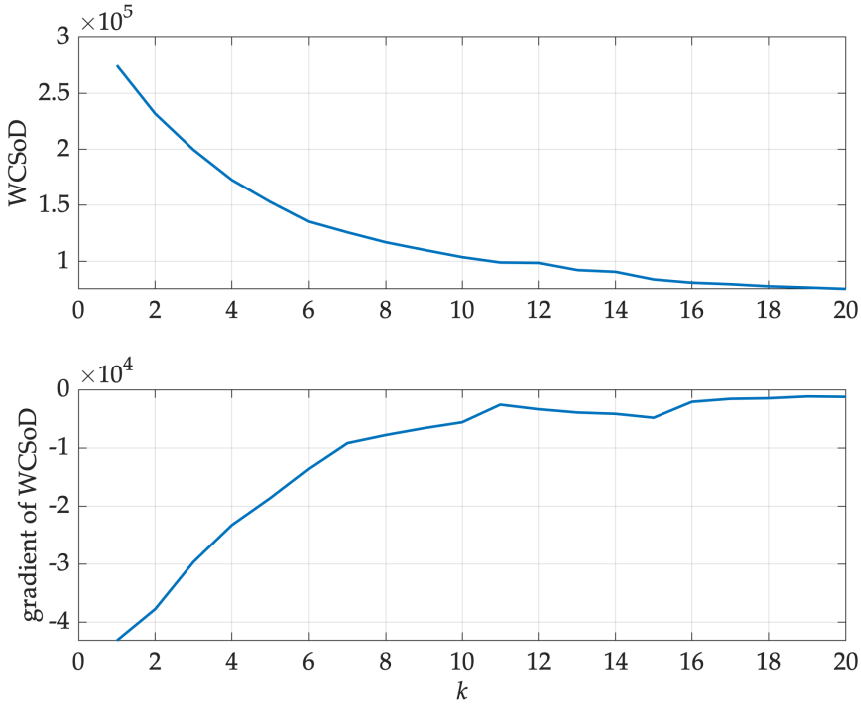
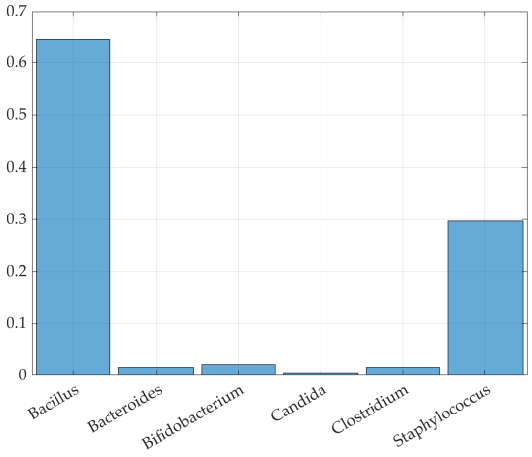
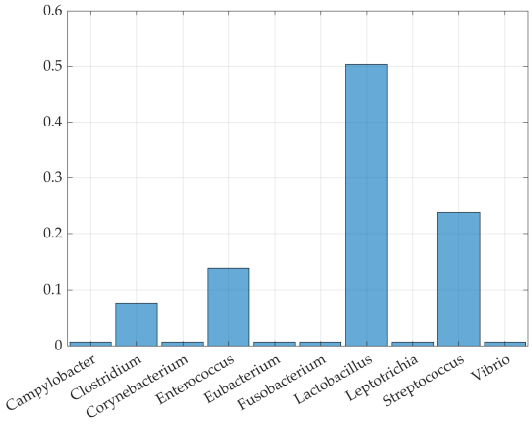


FIGURE 5.19: Gut flora organisms clustering, the elbow plot.

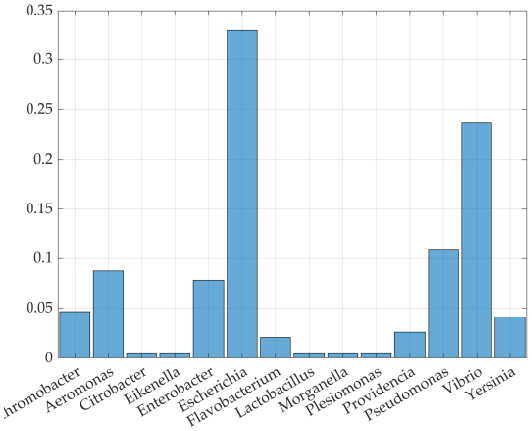
[45] This analysis leaded to a total number of 1380 edges, the complete listing is impossible.



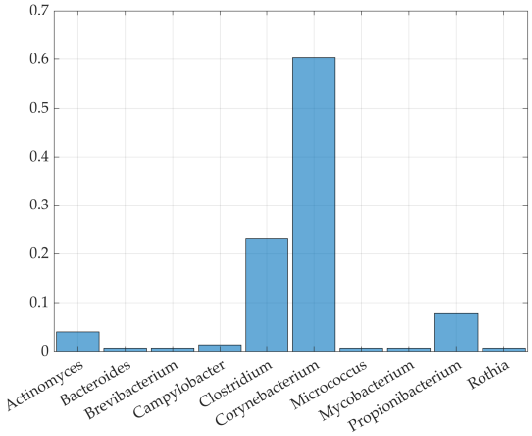
(A) Cluster 1



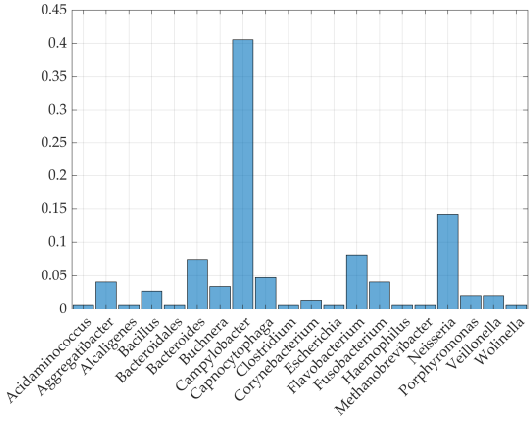
(B) Cluster 2



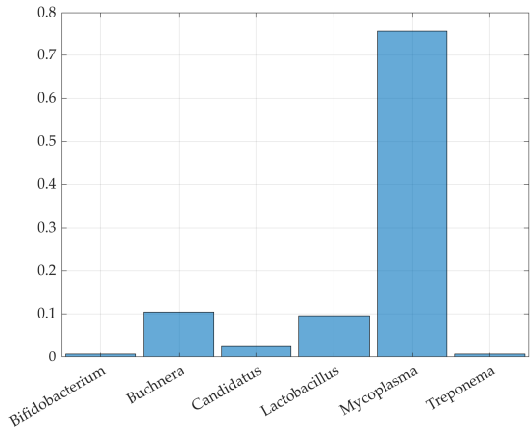
(C) Cluster 3



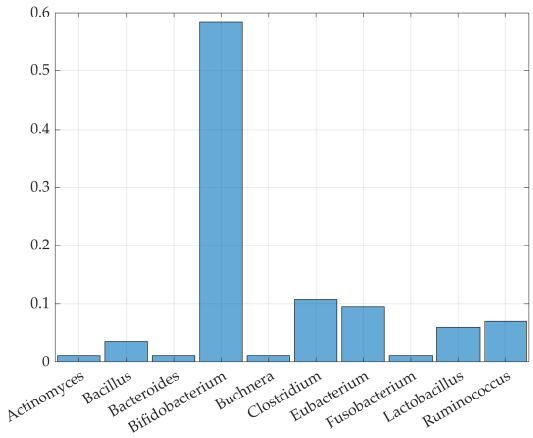
(D) Cluster 4



(E) Cluster 5



(F) Cluster 6



(G) Cluster 7

FIGURE 5.19: Gut flora organisms clustering, clusters composition.

TABLE 5.21: Gut flora organisms clustering, chemical reaction INDVAL scores. $I^{(a)}$ indicates the INDVAL score in cluster a. Source and target nodes have been identified according to their KEGG compound ID. In bold, the most relevant scores in each cluster.

Source	**Target**	$I^{(1)}$	$I^{(2)}$	$I^{(3)}$	$I^{(4)}$	$I^{(5)}$	$I^{(6)}$	$I^{(7)}$
cpd:C06552	cpd:C06553	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
cpd:C05893	cpd:C17550	0.02	**0.93**	0.00	0.00	0.00	0.02	0.02
cpd:C06364	cpd:C06365	0.00	**0.91**	0.00	0.03	0.00	0.00	0.06
cpd:C00423	cpd:C00079	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00
cpd:C00024	cpd:C00083	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00
cpd:C00173	cpd:C00170	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00
cpd:C20246	cpd:C20247	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
cpd:C00381	cpd:C00110	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
cpd:C00588	cpd:C00114	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
cpd:C01227	cpd:C00280	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
cpd:C00603	cpd:C00048	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
cpd:C00603	cpd:C00014	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
cpd:C00062	cpd:C00581	0.00	0.00	0.07	**0.93**	0.00	0.00	0.00
cpd:C00032	cpd:C00500	0.00	0.08	0.00	**0.92**	0.00	0.00	0.00
cpd:C00346	cpd:C00084	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
cpd:C00195	cpd:C02686	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
cpd:C18911	cpd:C03492	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
cpd:C02985	cpd:C01019	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**
cpd:C05557	cpd:C06564	0.00	0.00	0.00	0.00	0.10	0.00	**0.90**
cpd:C15858	cpd:C05432	0.00	0.67	0.00	0.00	0.00	**0.33**	0.00
cpd:C00455	cpd:C00153	0.10	0.03	0.16	0.06	0.31	**0.27**	0.06

The biochemical profiling offered by the INDVAL incredibly enriches the cluster analysis results, highlighting the common peculiarities amongst micro-organisms within the same cluster.

5.4.2 Metabolic Networks classification via INDVAL score

The metabolic networks clustering experiment paved the way towards the development of the INDVAL-based embedding strategy methodologically described in Section 4.4. The rationale behind the INDVAL-based technique can be summarised by the following question:

> Since the INDVAL spots relevant edges within the dataset, can it be used for evaluating meaningful information granules for embedding purposes?

The four problems (datasets) from Section 5.1.3 have been individually considered. Experiments herein presented can be found in [260]. In a first experiment, since the network formalisation and the Hamming distance have been proved to be effective in metabolic networks discrimination, the four problems have been solved by means of a plain K-NN decision rule equipped with the Hamming distance.

The model synthesis for this first experiment can be summarised as follows:

1. the dataset has been split into Training, Validation and Test Set, according the usual stratified sampling
2. for each K candidate in range $[1, 20]$, the K-NN exploits the Training Set in order to classify the Validation Set: the best K, say $K^\star$, is the one that maximises the informedness on the Validation Set. In case of multiple K's leading to the same results, the lowest is retained.
3. the final performances are evaluated on the Test Set, by exploiting $K^\star$ neighbours.

Tables 5.22–5.25 show the average performances on the Test Set for this first experiment. For the sake of completeness, the optimal $K^\star$ is also shown. Classes are numbered according to their order in Figure 5.4 (x-axis, left to right).

TABLE 5.22: Metabolic Networks classification, Problem 1 with K-NN, average performances on the Test Set.

ACC	**SNS**	**SPC**	**NPV**	**PPV**	$K^\star$
99.98	99.82	100	99.98	100	1

TABLE 5.23: Metabolic Networks classification, Problem 2 with K-NN, average performances on the Test Set.

Class	**ACC**	**SNS**	**SPC**	**NPV**	**PPV**	$K^\star$
1	99.95	100	99.95	100	98.46	1
2	99.89	98.18	99.98	99.90	99.55	1
3	99.80	99.99	98.51	99.94	99.77	1
4	99.91	98.39	99.95	99.96	97.85	1
5	99.99	100	99.99	100	99.57	1
6	99.83	81.67	100	99.83	100	1

TABLE 5.24: Metabolic Networks classification, Problem 3 with K-NN, average performances on the Test Set.

Class	ACC	SNS	SPC	NPV	PPV	$K^\star$
1	100	100	100	100	100	1
2	98.57	91.67	100	98.33	100	1
3	100	100	100	100	100	1
4	100	100	100	100	100	1
5	98.57	100	98.48	100	83.33	1

TABLE 5.25: Metabolic Networks classification, Problem 4 with K-NN, average performances on the Test Set.

Class	ACC	SNS	SPC	NPV	PPV	$K^\star$
1	100	100	100	100	100	1
2	100	100	100	100	100	1
3	100	100	100	100	100	1
4	99.97	100	99.97	100	99.41	1
5	100	100	100	100	100	1
6	100	100	100	100	100	1
7	99.89	100	99.88	100	98.43	1
8	100	100	100	100	100	1
9	99.97	99.44	100	99.97	100	1
10	99.97	99.55	100	99.97	100	1
11	99.89	98.46	100	99.88	100	1
12	100	100	100	100	100	1
13	100	100	100	100	100	1
14	100	100	100	100	100	1
15	100	100	100	100	100	1
16	99.97	100	99.97	100	99.72	1
17	100	100	100	100	100	1

In a second experiment, the INDVAL-based embedding space has been used in lieu of the entire networks, as in the K-NN case. For the sake of comparison, the very same Training-Validation-Test splits have been used.

The construction of the embedding space for this second experiment can be summarised as follows:

1. all patterns belonging to Training and Validation Set have been considered for building the embedding space
2. the set of unique edges (cf. $\overline{\mathcal{E}}$, Section 4.4) drawn from graphs belonging to the union of Training and Validation Set has been considered: each edge can be unambiguously identified by the (categorical) node labels at its extremities (namely, the KEGG IDs as in Table 5.21). In this manner, the evaluation of the unique edges is straightforward
3. for each edge, its INDVAL with respect to the problem-related classes has been evaluated thanks to Eqs. (4.16)–(4.18)

4. given a user-defined threshold T, all edges whose INDVAL score is greater than or equal to T for at least one of the problem-related classes becomes part of the alphabet $\mathcal{A}$ of candidate information granules

5. each graph from Training Set, Validation Set and Test Set is individually transformed into an $|\mathcal{A}|$-length vector enumerating the number of occurrences of each information granule (relevant edge) within the original graph (i.e., the symbolic histogram).

As per the embedding over simplicial complexes (see Section 5.2.3 for the EC number classification and/or Section 5.3.4 for the solubility classiication) both ν-SVMs and ℓ_1-SVMs are used for classification. The two classification models are driven by a genetic algorithm which sees a genetic code of the form as in Eq. (5.6) for ν-SVMs and a genetic code that reads as in Eq. (5.8) for ℓ_1-SVMs. For both cases, the fitness function reads as in Eq. (5.7) (with the hyperplane coefficient vector in lieu of the explicit feature selector $\mathbf{w}$ in the ℓ_1-SVMs case). For both SVMs, the fitness function tradeoff parameter α has been set as $\alpha = 0.5$ in order to give the same importance to sparsity and performances. Since the INDVAL is in range $[0, 100]$, a reasonable threshold $T = 50$ is selected for prior edges filtering and alphabet synthesis.

Table 5.26 shows the starting alphabet size for all four problems, averaged across the several Training-Validation-Test splits.

TABLE 5.26: Metabolic Networks classification, average size of the starting alphabet for building the INDVAL-based embedding space ($T = 50$).

Problem	1	2	3	4
Alphabet Size	621	825	234	98

Tables 5.27–5.30 show the average performances across five Training-Validation-Test Set splits on the Test Set for the four problems. As per the embedding over simplicial complexes case, the sparsity is also shown. In bold, the best between the two SVM-based models.

TABLE 5.27: Metabolic Networks classification, Problem 1 via INDVAL-based embedding, average performances on the Test Set ($T = 50$).

Classifier	**ACC**	**SNS**	**SPC**	**NPV**	**PPV**	**Sparsity**
ℓ_1-SVM	**99.98**	**99.82**	**100**	**99.98**	**100**	**2.03**
ν-SVM	**99.98**	99.73	**100**	**99.98**	**100**	21.9

Amongst the two SVMs-based systems, ℓ_1-SVMs overperform ν-SVMs in terms of alphabet selection in the vast majority of cases (Problems 1–3). The opposite is true only for few classes from Problem 4. Apart from a minor shift in sensitivity, the two systems perform equally for Problem 1 (Table 5.27). For Problems 2 and 3, ℓ_1-SVMs overperform ν-SVMs in terms of sensitivity and negative predictive value; the opposite is true for specificity and positive predictive value. For Problem 4, classes 4, 7, 9 and 12 seem harder to discriminate given the low precision (PPV) and low sensitivity (for ℓ_1-SVMs). It is possible to improve these results by changing the threshold T, hence Problem 4 has been solved again by lowering the threshold from $T = 50$ to $T = 30$. Table 5.31 shows the corresponding results where, for the sake of shorthand, only classes 4, 7, 9 and 12 are displayed.

TABLE 5.28: Metabolic Networks classification, Problem 2 via INDVAL-based embedding, average performances on the Test Set ($T = 50$).

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
1	ℓ_1-SVM	99.86	**100**	99.86	**100**	96.11	**0.67**
	ν-SVM	**99.98**	99.55	**100**	99.98	**100**	26.36
2	ℓ_1-SVM	**99.87**	**99.55**	99.89	**99.98**	97.98	**1.52**
	ν-SVM	**99.87**	97.88	**99.98**	99.89	**99.55**	25.86
3	ℓ_1-SVM	98.86	98.82	**99.14**	92.79	**99.87**	**2.77**
	ν-SVM	**99.83**	**99.97**	98.86	**99.83**	99.83	22.41
4	ℓ_1-SVM	98.75	**99.35**	98.74	**99.98**	67.25	**1.64**
	ν-SVM	**99.92**	96.77	**100**	99.92	**100**	28.39
5	ℓ_1-SVM	99.86	**98.64**	99.88	**99.98**	93.29	**0.6**
	ν-SVM	**99.95**	97.27	**100**	99.95	**100**	34.02
6	ℓ_1-SVM	99.41	**85**	99.54	**99.86**	65.27	**3.27**
	ν-SVM	**99.81**	83.33	**99.96**	99.85	**95.69**	31.33

TABLE 5.29: Metabolic Networks classification, Problem 3 via INDVAL-based embedding, average performances on the Test Set ($T = 50$).

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
1	ℓ_1-SVM	**98.86**	**95**	99.35	**99.36**	95.5	**3.34**
	ν-SVM	**98.86**	90	**100**	98.79	**100**	20.07
2	ℓ_1-SVM	**100**	**100**	**100**	**100**	**100**	**0.43**
	ν-SVM	97.14	85	99.66	97.05	98.33	17.44
3	ℓ_1-SVM	**99.71**	**99**	**100**	**99.62**	**100**	**2.92**
	ν-SVM	98.86	96	**100**	98.49	**100**	7.4
4	ℓ_1-SVM	**100**	**100**	**100**	**100**	**100**	**0.89**
	ν-SVM	96.86	92.31	99.55	95.83	99.17	8.29
5	ℓ_1-SVM	97.14	**90**	97.58	**99.36**	81.67	**4.82**
	ν-SVM	**98**	80	**99.09**	98.81	**88.33**	15.3

As can be easily seen by matching Tables 5.22–5.25 with Tables 5.27–5.31, *K*-NN slightly outperforms the two SVMs-based classification systems, both relying on the proposed information granulation and embedding. However, the latter approach has an overwhelming advantage, since it enables knowledge discovery, of utmost importance in most data driven modelling applications. Specifically, the *K*-NN serves as a suitable benchmark for the following two facets:

1. *K*-NN is the simplest decision rule, with no training procedure, which does not return any compressed model of the system to be analysed

2. *K*-NN exploits the entire metabolic networks.

In other words, *K*-NN exploits the entire information contained in each pattern of the training set. The proposed technique, as instead, relies on an embedding procedure on the top of relevant edges of the considered networks (instead of the entire network), with a further feature selection phase that allows to select a suitable subset of such edges. Since the performances are rather comparable, the proposed embedding procedure retains the vast majority of the information contained in the original

dataset. This adds up to the aforementioned knowledge discovery phase, totally absent in the K-NN experiment, as new predictions are made by plain matching of the adjacency matrices. Between the two SVMs-based systems, the same observations discussed at the end of Section 5.2.3 still hold: ℓ_1-SVMs are faster and easier to train due to the underlying linear (rather than a quadratic) programming problem, their optimisation phase is easier due to the intrinsic feature selection phase.

TABLE 5.30: Metabolic Networks classification, Problem 4 via INDVAL-based embedding, average performances on the Test Set ($T = 50$).

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
1	ℓ_1-SVM	97.15	**95.31**	97.33	**99.55**	83.14	**3.46**
	ν-SVM	**99.31**	93.75	**99.85**	99.4	**98.46**	5.77
2	ℓ_1-SVM	95.6	**70.77**	96.52	98.88	70.08	8.65
	ν-SVM	**98.95**	**70.77**	**100**	**98.92**	**100**	**7.24**
3	ℓ_1-SVM	**99.89**	**100**	**99.88**	**100**	**98.3**	**3.5**
	ν-SVM	99.22	97.73	99.32	99.85	90.86	4.2
4	ℓ_1-SVM	**89.2**	17.5	**92.52**	96.02	16.37	**2.12**
	ν-SVM	80.08	**78.13**	80.17	**98.92**	**25.27**	36.51
5	ℓ_1-SVM	98.86	**100**	98.77	**100**	87.5	**1.02**
	ν-SVM	**99**	**100**	**98.92**	**100**	**89.03**	3.68
6	ℓ_1-SVM	95.04	89.17	95.24	99.62	59.23	**6.56**
	ν-SVM	**99.06**	**90**	**99.37**	**99.66**	**84.9**	9.6
7	ℓ_1-SVM	**86.57**	15.83	**91.6**	93.85	**23.94**	**5.2**
	ν-SVM	77.06	**94.17**	75.85	**99.45**	22.33	27.54
8	ℓ_1-SVM	96.54	**99.41**	96.4	**99.97**	60.8	**2.54**
	ν-SVM	**99.67**	98.24	**99.74**	99.91	**95.09**	6.08
9	ℓ_1-SVM	**90.11**	4.44	**94.61**	94.96	6.22	**2.5**
	ν-SVM	82.69	**76.67**	83	**98.78**	**15.64**	40.88
10	ℓ_1-SVM	93.52	**99.55**	93.13	**99.97**	58.25	5.63
	ν-SVM	**98.86**	98.64	**98.88**	99.91	**86.16**	**4.69**
11	ℓ_1-SVM	99.56	**98.85**	99.61	**99.91**	95.51	4.11
	ν-SVM	**99.7**	97.69	**99.85**	99.82	**98.25**	**2.66**
12	ℓ_1-SVM	22.3	1.36	23.66	73.16	0.1	89.89
	ν-SVM	**67.15**	**99.09**	**65.07**	**99.92**	**20.7**	**50.31**
13	ℓ_1-SVM	**99.78**	**100**	**99.76**	**100**	**97.01**	**1.02**
	ν-SVM	99.72	99.6	99.73	99.97	96.7	4.91
14	ℓ_1-SVM	96.43	94.62	96.49	99.8	58.34	**1.33**
	ν-SVM	**99.83**	**96.92**	**99.94**	**99.89**	**98.57**	5.11
15	ℓ_1-SVM	**99.94**	**100**	99.94	**100**	99.09	**1.42**
	ν-SVM	**99.94**	99.05	**100**	99.94	**100**	4.79
16	ℓ_1-SVM	97.04	**98.57**	96.87	99.84	81.65	8.23
	ν-SVM	**99.72**	**98.57**	**99.85**	**99.85**	**98.59**	**4.27**
17	ℓ_1-SVM	**100**	**100**	**100**	**100**	**100**	**1.02**
	ν-SVM	99.56	92	99.88	99.66	97.23	7.83

On the Impact of the Threshold T

The choice of T, albeit being the only parameter to be tuned by the end-user in order to build the alphabet, is critical: a higher T generally leads to a smaller alphabet

TABLE 5.31: Metabolic Networks classification, Problem 4 via INDVAL-based embedding, average performances on the Test Set ($T = 30$)

Class	Classifier	ACC	SNS	SPC	NPV	PPV	Sparsity
4	ℓ_1-SVM	91.97	**100**	91.59	**100**	40.08	**1.73**
	ν-SVM	**99.31**	93.13	**99.59**	99.68	**92.32**	11.51
7	ℓ_1-SVM	98.14	**97.5**	98.19	**99.82**	83.07	**5.75**
	ν-SVM	**99.72**	96.67	**99.94**	99.76	**99.23**	16.24
9	ℓ_1-SVM	92.44	**100**	92.04	**100**	44.19	**2.24**
	ν-SVM	**99.34**	94.44	**99.59**	99.71	**94.43**	11.11
12	ℓ_1-SVM	95.37	48.18	98.44	96.69	72.32	**9.92**
	ν-SVM	**99.14**	**97.27**	**99.26**	**99.82**	**90.39**	10.92

which, in turn, leads to a smaller embedding space which might not quite properly describe the original input space. Conversely, a larger embedding space might be computationally expensive, especially when it comes to select a suitable subset of meaningful alphabet symbols. Indeed:

- the distance between two vectors in $\mathbb{R}^m$, either evaluated as a plain Euclidean distance or via dot product, has complexity $\mathcal{O}(m)$
- in case of (evolutionary) metaheuristics, the search space is strictly related to the alphabet size: a larger alphabet implies a larger search space which, in turn, might lead to a larger number of individuals/generations (for better exploration) and/or larger probability of being trapped in local minima.

The aim of this Section is to investigate the impact of the threshold T in terms of embedding space complexity (i.e. dimensionality) against performances on the test set, where $T \in [10, 90]$ with step size 10. The four Problems and corresponding splits being equal, Figures 5.20–5.23 shows the tradeoff between alphabet size (both before – after T thresholding – and after the feature selection phase) and the performances in terms of (normalised) informedness when using ℓ_1-SVMs as classification system. From Figures 5.20–5.23, it is clear that, as the problem gets more difficult, more edges are needed (e.g., a smaller T). Indeed:

1. eukaryotes and prokaryotes can be (almost) perfectly classified even at $T = 90$ (the largest tested value)
2. different kingdoms can be (almost) perfectly classified for $T \leq 80$: note from Figure 5.21 that the informedness for class 6 drops from circa 90% to circa 60% when sliding T from 80 to 90
3. organisms within the animal kingdom can be (almost) perfectly classified for $T \leq 60$: see from Figure 5.22 the behaviour of classes 1 and 5 for $T > 60$
4. different bacteria can be (almost) perfectly classified for $T \leq 30$: only classes 1, 3, 5, 11, 13, 15 and 17 show outstanding classification performances up to $T = 90$.

Similar considerations hold when using ν-SVMs as classification systems, as reported in Figures 5.20–5.23, which are, however, featured by a higher number of selected symbols (as already shown in Tables 5.27–5.31).

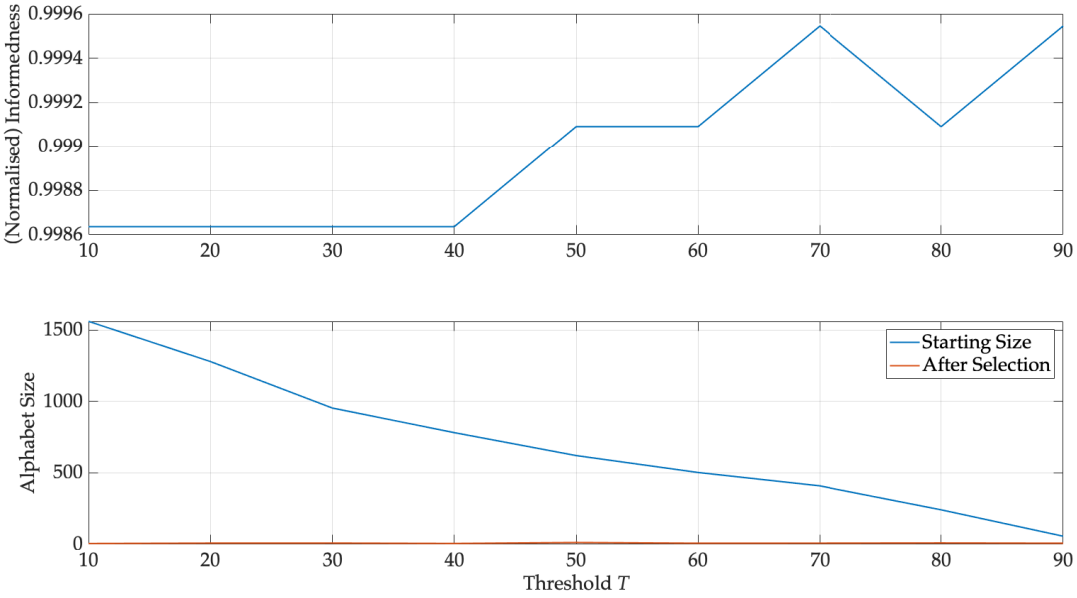


FIGURE 5.20: Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 1).

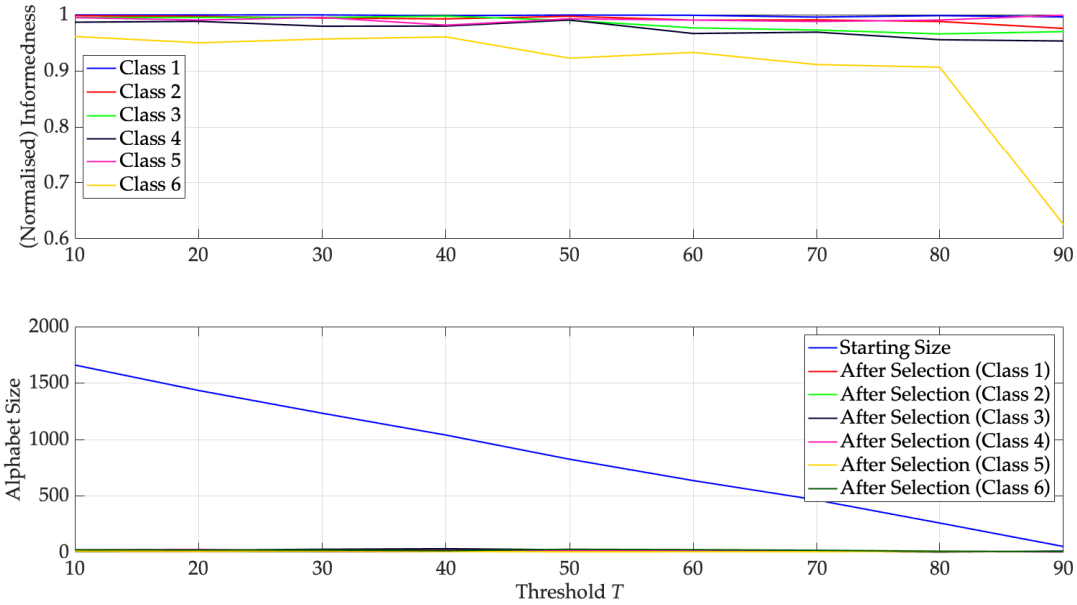


FIGURE 5.21: Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 2).

5.4.3 Statistical Assessment of Classification Results

In order to statistically show the difference in classification, the benchmark against the *dummy classifier* (also, *blind classifier*) is investigated [113, 249, 260]. The blind classifier randomly generates the output values (labels) just by considering their statistics within the dataset. If the classification problem has N_L labels, then the probability to output the i^{th} label is given by:

$$p(l_i) = \frac{|\mathcal{D}_{\text{TR}}^{(l_i)}|}{|\mathcal{D}_{\text{TR}}|} \qquad \forall i = 1, \ldots, N_L \tag{5.19}$$

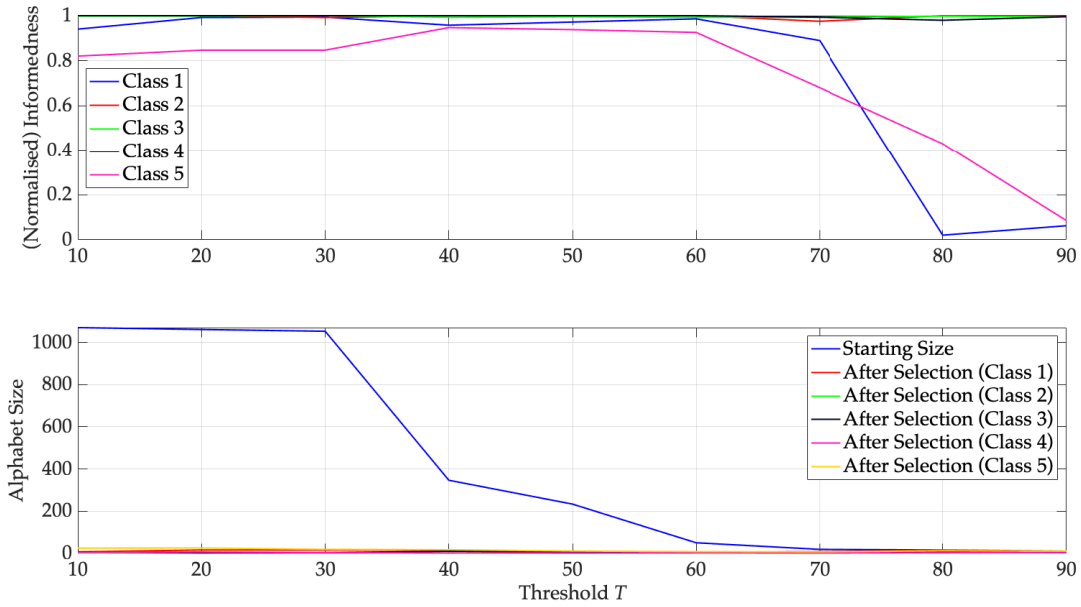


FIGURE 5.22: Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 3).

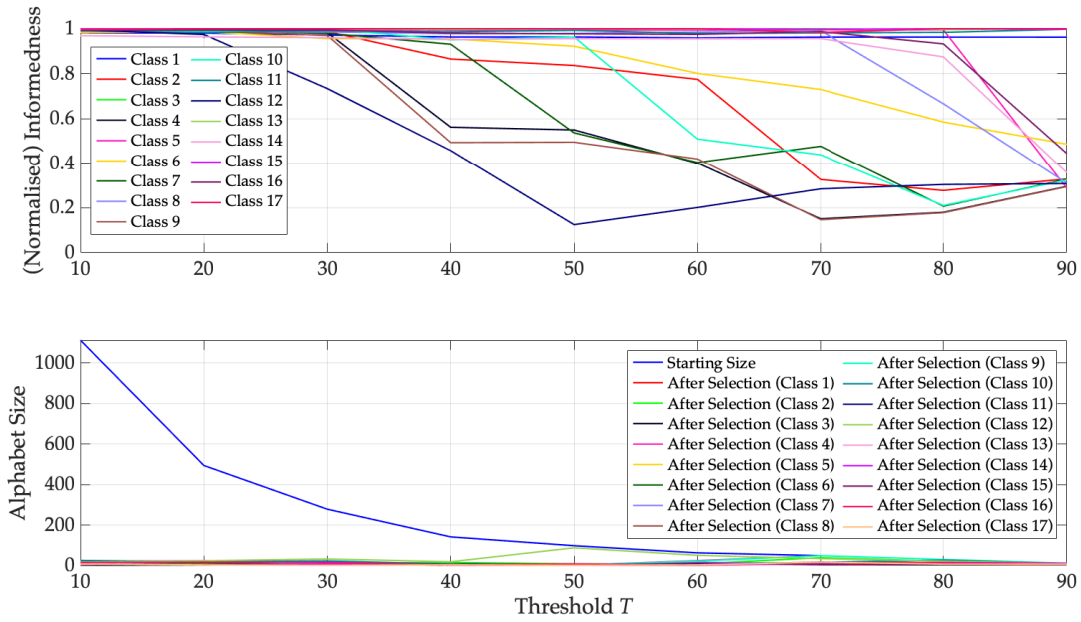


FIGURE 5.23: Metabolic Networks classification, average performances and alphabet size as function of T (ℓ_1-SVMs, Problem 4).

where $|\mathcal{D}_{\text{TR}}^{(l_i)}|$ is the number of patterns in the training set having label l_i and $|\mathcal{D}_{\text{TR}}|$ is the cardinality of the training set.

The blind classifier, by definition, does not consider the information carried out by the training data: the 'blindness' refers to the fact that in order to assign the output label to a given patter, say $\mathbf{x}$, the statistical inference adopted by the blind classifier does not consider in any way the information carried out by $\mathbf{x}$ itself. Hence, the blind classifier can dually quantify the amount of information that the dataset contains and serve as null-model for the considered classification systems, either based on the entire network topology (K-NN) or on the INDVAL-based strategy (ℓ_1-SVM and ν-SVM).

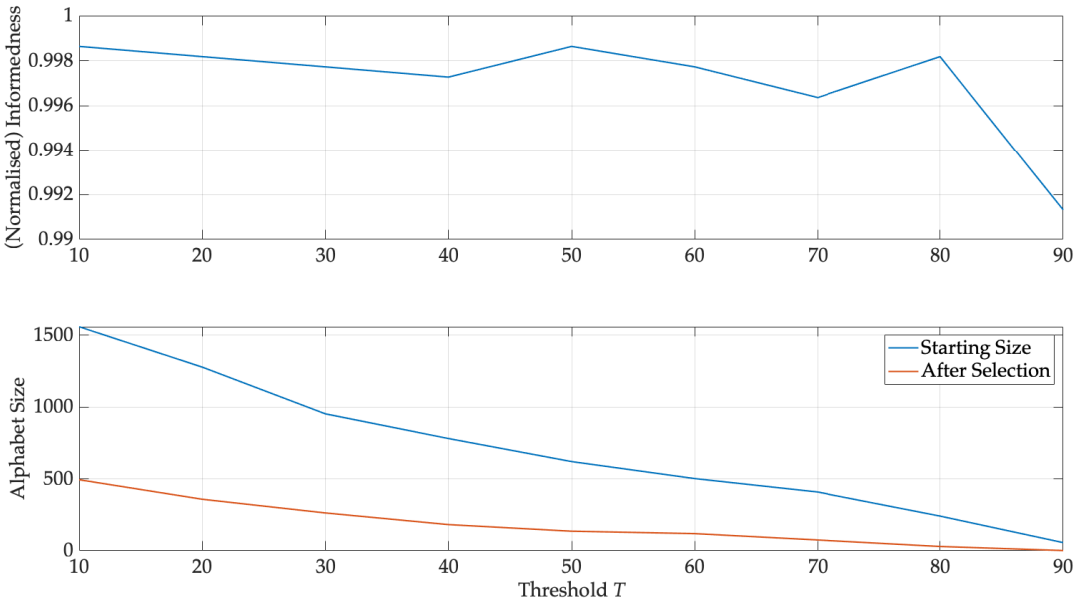


FIGURE 5.24: Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 1).

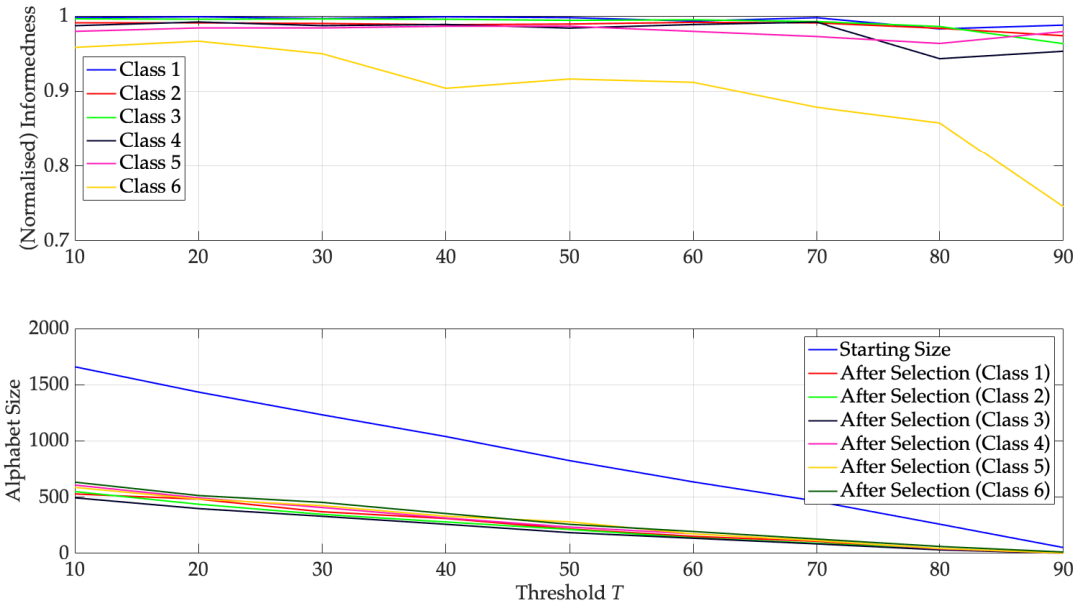


FIGURE 5.25: Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 2).

Figure 5.28 shows via box plots[46] the (normalised) informedness of 1000 blind classifiers against the three proposed classification systems. For the sake of comparison with the tables in the main manuscript, results for ν-SVMs and ℓ_1-SVMs have been obtained by considering $T = 50$.

For Problems 1–3 (i.e., Figures 5.28a–5.28c) the statistical validity is striking. For Problem 4 (Figure 5.28d) it is possible to see the weird behaviour for classes 4, 7, 9 and 12: ν-SVMs tend to deteriorate (average informedness of approx. 0.8). On the

[46] Red horizontal markers indicate the median value; boxes top and bottom correspond to 75th and 25th percentiles; top and bottom whiskers correspond to the maximum and minimum not-outliers; red plus signs correspond to outliers.

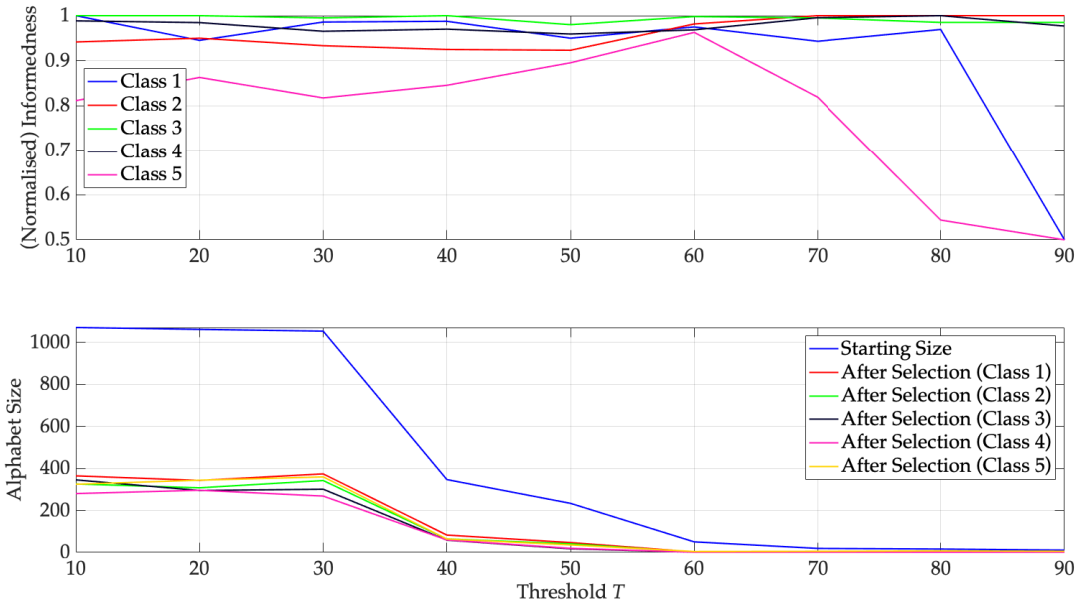


FIGURE 5.26: Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 3).

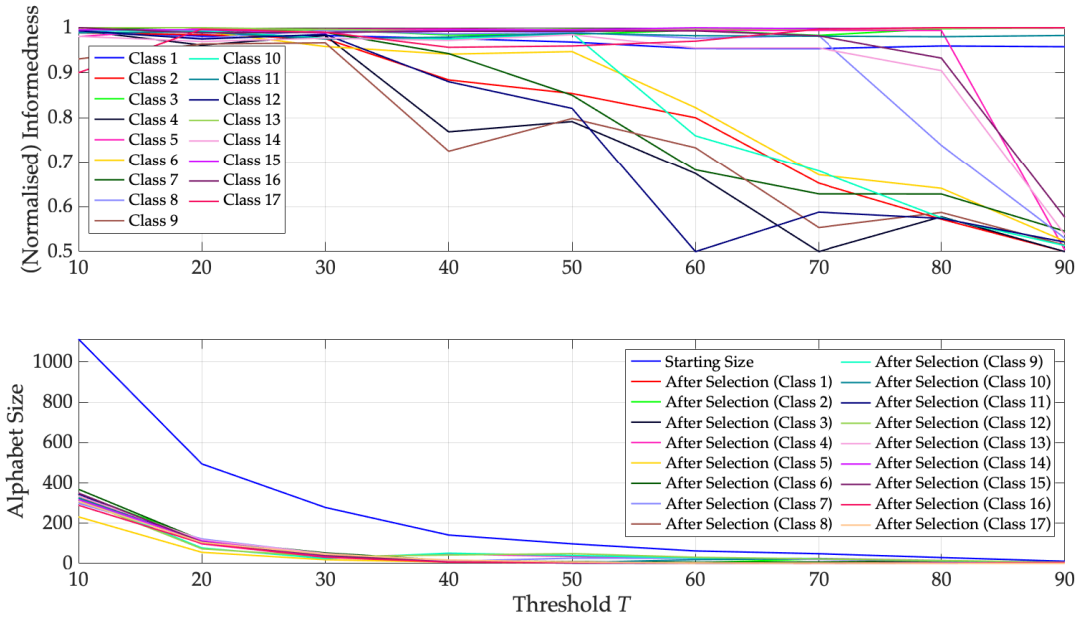


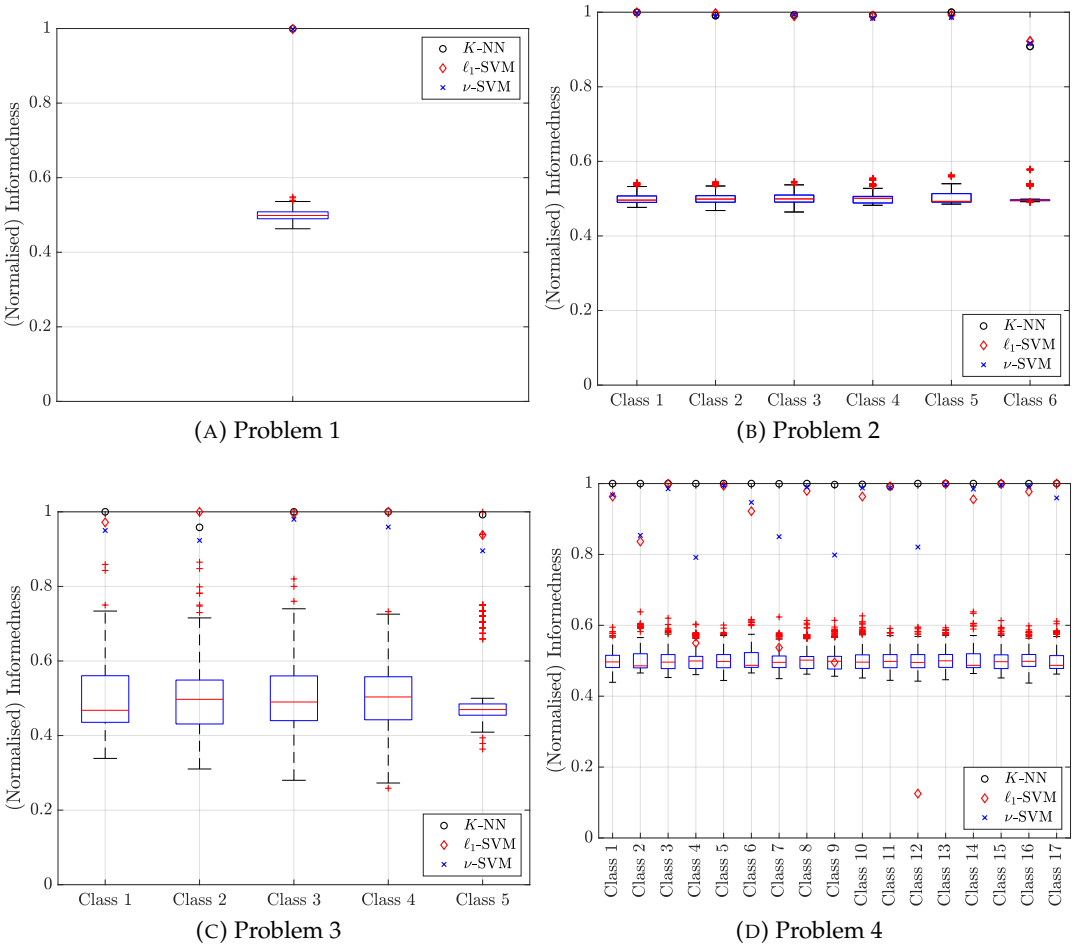
FIGURE 5.27: Metabolic Networks classification, average performances and alphabet size as function of T (ν-SVMs, Problem 4).

other hand, ℓ_1-SVMs seem to have the same performance as the blind classifier for classes 4, 7, 9 and seem to perform worst than the dummy classifier for class 12. This is perfectly in line with the results presented in Table 5.30, in which we discussed that these classes cannot be linearly separated at $T = 50$ and a lower threshold is needed.

For blind classifiers, the informedness is always approximately 0.5[47]: this is perfectly coherent with the rationale behind the definition of *informedness*. As discussed

[47]That is because we adopted the normalised informedness, otherwise the informedness for pseudo-random classifiers is approximately 0.

in [316], the informedness can be interpreted as the probability of an informed decision rather than a random guess. By excluding the aforementioned 'problematic' classes in Problem 4 (that is because of the high value $T = 50$, cf. Table 5.30 and 5.31) for ℓ_1-SVMs, all classifiers have informedness very close to 1 (its maximum value): another confirmation of the statistical validity of the proposed classification systems.



(A) Problem 1 (B) Problem 2

(C) Problem 3 (D) Problem 4

FIGURE 5.28: Performance of the blind classifier for all four metabolic pathways problems.

5.4.4 Final Remarks

Besides the efficiency in classification, the INDVAL strategy allows the field-expert to focus on specific substructures (edges, in this work) at the basis of the embedding procedure, namely substructures marked as 'important' for the classification system. Indeed, it is worth stressing that the threshold T leads to a prior substructures filtering, whereas the feature selection phase during classification refines the alphabet selection.

As far as the metabolic networks application is concerned, the final set of biochemical reactions can give some interesting hints for the functional basis of the among organisms differences. All of the four problems gave rise to interesting biological solutions in terms of INDVAL 'almost perfect' edges (score non-zero for a given class vs. score 0 for other classes) or 'perfect' edges (score 100 for a given class vs. score 0 for all other classes), which will be separately discussed in the following:

Problem 1. As reported in Table 5.32, all the eukaryotic-only edges involve (directly or indirectly) Inositol metabolism. The puzzling point is that Inositol molecule is present in prokaryotes too, but while in prokaryotes Inositol is mainly involved in maintaining a correct acid-base balance protecting some bacterial species from inactivation by the host during infections, in eukaryotes via both its antioxidant activity and its ability to detoxify a variety of toxic thiol-reactive compounds [326] this molecule (in the form of inositol phosphate) is a key component of (eukaryotic) signalling pathways that do not occur in bacteria and archaea [245]. Biological membranes (both nuclear and cell membrane) are the 'gates' mediating these signalling pathways and all the 'Inositol-linked' signature edges have to do with reactions involving Inositol in conjunction with membrane components.

The most prominent difference between prokaryotic and eukaryotic organisation (reflected by their respective Greek-derived names: *eu-karios* means 'well formed nucleus' while *pro-karios* translates into 'primitive nucleus') is the absence in prokaryotes of a definite compartmentalisation between nucleus (the site of genetic information) and the rest of the cell, while this compartmentalisation is very strict in eukaryotes [1]. Besides the nucleus/cytoplasm separation eukaryotic cells display a very ordered and complex patterning of different compartments (organelles) that allow to keep separate different cell functionalities. Such a fundamental organisation difference between eukaryotic and prokaryotic cells could give the (false) impression that there are many metabolic pathways unique for each group, spanning the entire range of functionalities. This is not the case, the ecological dimension of life rules out any too simplistic quantitative reasoning: bacteria play a crucial role in both external (rivers, lakes, soil, oceans) and internal (gut, nose, buccal and vaginal mucosa) ecologies in terms of degradation and production of biological matter. This role implies that the products of bacterial metabolism must 'match' with enzymes able to transform these metabolites so to enter into eukaryotic metabolism while at the same time, to accomplish the task of recycling the degradation products coming from eukaryotic forms, the bacterial metabolism must have the specific enzymes to let these products to enter prokaryotic metabolism.

This 'recycling' activity is essential for the maintaining of life (by the way, bacteria constitute a consistent part of the global biomass [24]) and necessitates the establishing of strong functional relations between such incredibly distant kingdoms of life. The intermingled relation between evolutionary and environmental dimensions at the metabolic level is at the basis of the utmost interest of metabolic network wiring comparison.

On the other side, the fact that only 'unique edges' are linked to inositol metabolism, is of utmost interest for shedding light on the role of inositol in 'purely multicellular/eukaryotic' features like differentiation and cancer.

The eukaryotic organisation makes the membranes the preferred site of signalling between the external environment and cell allowing for cell differentiation (and thus the following development of specialised tissues and organs). This is why the appearing (after approximately two billion years of prokaryotic-only life) of the eukaryotic cell is considered by many scientists the most prominent milestone in the evolution of life [200].

The fact that such a complete re-shaping of cell architecture presents a 'perfectly discriminating signature' in terms of a relatively simple molecular system like Inositol (and its derivatives involved in membrane signalling) opens

very interesting perspectives. As for biological theory, this result suggests a single critical transition that, by a 'different use' of an already present molecule, triggers a cascade of events making possible superior forms of life by a sort of 'compartmentalisation revolution'. In more applicative terms, the (largely unexpected) 'fundative' role of Inositol helps to get rid of the accumulating evidence of the involvement of Inositol in different regulation circuits of utmost biomedical interest like psychiatric syndromes, diabetes, fertility problems and cancer [39, 241, 352].

Problem 2. Archaea are the last kingdom of the tree-of-life to be discovered, they are prokaryotes, like bacteria, and are considered the organisms most similar to the initial forms of life on Earth as well as the direct precursors of eukaryotic forms [248]. The chemical reactions that better allow archaea identification are linked to Acetyl-CoA metabolism and consequently to fatty acid metabolism. These arches are particularly relevant to distinguish archaea from eubacteria [117]. Acetate and Acetyl-CoA play fundamental roles in all of biology, including anaerobic prokaryotes from the both bacteria and archaea, which compose an estimated quarter of all living protoplasm in Earth's biosphere. Anaerobes from the domain Archaea (at odds with Bacteria) have the unique property to produce methane, so contributing to the global carbon cycle by metabolising acetate as a growth substrate or product. Acetate-utilising anaerobes classified in the domain Archaea also proliferate in environments where terminal electron acceptors are abundant and obtain energy through anaerobic respiration, converting acetate to CO_2 [138, 334]. The Archaea colonise extreme environments (e.g., thermal source at near 100° temperature or the bottom of the oceans at incredibly high pressure, ices, ...) so allowing for the global Earth carbon cycle. In this case, the metabolic network analysis allowed to catch the essential of the organisms ecology, complementing the classical genetics based phylogenesis with the peculiar environmental (phenotypic) role of these organisms.

Most of the Earth biomass is made by plants, this is not strange by considering that the primary input of matter into the biosphere comes from plants, which are responsible of the initial transformation of the primary energy input (light coming from the Sun) into nutrients that are then made accessible to other organisms. This singular position of plants at the beginning of global metabolic cycles is made possible by photosynthetic pigments (mainly chlorophyll) that are in charge to 'capture' photons. This is why this kingdom has a lot of 'specific edges' (unique metabolic reactions) mainly linked to the production and/or transformation of these pigments. Under the same heading, being plants 'primary producers' of organic material, some constituents have been found (e.g. mannose, oxaloacetate) that are peculiar to plants that in turn individuate specific reactions in which they are involved in the plant metabolic networks.

At odds with plants, that are autotroph (a Greek word that indicates that they produce by themselves their food), fungi are heterotroph (need an external source of food) and their metabolic peculiarities are strictly related to their 'struggle for life' in soil ecosystems where they must compete with different kind of organisms (mainly bacteria) to occupy their trophic niche. Human made use of the weapons used by fungi to find a place in the extremely competitive soil ecology: the most known example is penicillin, an antibiotic molecule

produced by the fungi of the *Penicillium* class to eliminate bacteria competing for the same habitat. In general, mushrooms are recognised as valuable sources of natural products with a great structural diversity, including cyclic peptides, steroids, sesquiterpenes and polysaccharides. They are known to exhibit various beneficial pharmacological properties such as antibacterial, immunomodulatory, hypocholesterolemic and antioxidant activities with a considerable therapeutic potential [373]. In our analysis, the fungi maximum INDVAL specific edges are related to the metabolism of ergostane and its derivatives: these molecules are hormones endowed with a marked biological activity going from antimicrobial to antioxidant and anti-inflammatory effects that are now under the focus of pharmacological research [61, 123]. The unique features (discovered by the INDVAL-based approach) of ergostane derivatives suggest they could be a still unexplored source of new drugs.
The metabolic reactions unique to animals involve myoinositol metabolism and the reaction converting phosphatidylcholine into phosphatidyl serine. The crucial role of myoinositol when talking about the eukaryotic/prokaryotic discrimination has already been discussed for Problem 1 (see Table 5.32). This result allows us to better focus on the emerging role myoinositol that is not generically 'eukaryotic-like' but 'animal-like', so reinforcing the unique role of membrane signalling in animals with respect to other kingdoms of life. This result is consistent with the 'phosphatidylcholine' metabolism. Phosphatidylcholine is a major constituent of cell membranes and pulmonary surfactant, and is more commonly found in the exoplasmic or outer leaflet of a cell membrane. Phosphatidylcholine also plays a role in membrane-mediated cell signalling and activation of enzymes [189].
Summing up the biological evidences coming from the 'kingdom of life' discrimination is evident how the 'environment-organism' interaction is managed at different levels of organism complexity. At lower level (Archaea) the single cell must cope with external (mainly physical) environment and the 'metabolic peculiarities' (high INDVAL edges) reflect the survival strategy in extreme situations. Going up the complexity scale (Fungi), competing organisms become a crucial part of the environment that in turn becomes 'more biological' and asks for specific 'chemical weapons' to survive. At the very end of organisation complexity (Animals) the single cell has no role to play in terms of environmental interactions: in a complex multicellular organism the main issue is to maintain an efficient signalling among cells so they can work in a coordinated way. The 'environment' of each cell are the other cells and coping with external environment is a matter of organised aggregates of cells like tissue, organs and integrated systems (e.g., immune system, nervous system, cardiovascular system, ...). This is registered by the animals specific edges that are all related to among cells signalling. All relevant edges discussed so far for Problem 2 are summarised in Table 5.33.

Problem 3. As far as Problem 3 is concerned, ℓ_1-SVMs selected a single, 'perfect' edge able to perfectly discriminate fishes against non-fishes. In other words, by using ℓ_1-SVMs, in order to check (with 100% accuracy, see Table 5.29) whether a given organism is a fish or not, it is possible to check whether this single edge (chemical reaction) exists or not in its metabolic network. The very same edge has also been selected by ν-SVMs, however alongside other edges. This means that ℓ_1-SVMs have been well able to exploit the 'perfection' and 'meaningfulness' of this edge for perfect classification: indeed, the INDVAL values for all

other classes (birds, insects, mammals and reptiles) is 0 whereas the INDVAL for fishes is 100. This 'perfect' chemical reaction is involved in ω-3 fatty acids biosynthesis and is reported in Table 5.34. In fish species, lipids and proteins are the main organic constituents and play many important roles in the fish's life history and physiology, which includes growth, movement, reproduction and migration. The main role of lipids in these organisms is the storage and provision of energy in the form of Adenosine Triphosphate (ATP) through the β-oxidation of fatty acids [294]. Furthermore, the fatty acids of fish lipids are rich in ω-3 long chain, highly unsaturated fatty acids (n-3 HUFA) that have particularly important roles in animal nutrition, including fish and human nutrition, reflecting their roles in critical physiological processes. Indeed, fish are the most important food source of these vital nutrients for man. Thus, the long-standing interest in fish lipids stems from their abundance and their uniqueness [358]. This uniqueness stems from fatty acid biosynthesis pathways only present in fishes. These 'fish-only' pathways have the phosphorylation of D-Glucuronate as signature.

Problem 4. At odds with the other three Problems, in which the 'golden standard' biological classification can be considered relatively stable, the discrimination among different bacterial classes is still in a turmoil. A paper recently appeared on Nature Biotechnology [302] reveals that a recently introduced phylogenetic approach made the 58% of the 95789 bacterial genomes known until last year, to change their taxonomy. It is worth stressing the fact that, in biology, taxonomy (i.e., the position of an organism in the tree of life) is ideally based on evolutionary relationships among organisms. Development of a robust bacterial taxonomy has been hindered by an inability to obtain most bacteria in pure culture and, to a lesser extent, by the historical use of phenotypes to guide classification [157, 302]. Notwithstanding that, the emerging importance of the study of microbiota (i.e., the pattern of species or higher taxonomic aggregations) in human mucosa (mouth, gut, vagina, ...) in many pathologies makes the obtainment of a meaningful classification urgent. Beside the importance of an evolution based phylogeny, it is perhaps still more crucial to establish an ecological based taxonomy that not necessarily coincides with the evolutionary one (as stressed at the beginning of Section 5.4). This is a very common feature in the history of life: organisms that are not phylogenetically related can be very similar due to their common environmental needs. This phenomenon is called *evolutionary convergence* and is at the basis of well known 'only phenotypic similarities' like the features shared by dolphins and some fishes, due to the common needs of marine life, in spite of their lack of shared ancestry. In the case of the definition of an optimal microbial ecology of the gut (or any other mucosa) so to both appreciate disease-associated patterns and try to devise therapeutic strategies, a first step is to find an ecologically relevant phenotypic formalisation of bacteria. Metabolic networks are perfectly fit for this purpose for the simple fact the basic role played by the microbes in the gut is to carry out a metabolic work useful for the host (that can be turned into a damage in the case of an altered microbial ecology). Before entering this very ambitious issue, this experiment successfully dealt with a preliminary step in order to check the ability of the proposed approach to recognise 'established bacterial taxonomic classes'. It is worth noting that for very few exceptions (Chlamydia,

Pseudomonas, Staphylococcus and Streptomyces) the INDAVAL-based strategy was able to find a unique diagnostic edge univocally classifying these bacterial classes. On the other hand, for other classes (Campylobacter, Corynebacterium, Helicobacter and Mycoplasma) there are no relevant edges (coherently with the results presented in Table 5.30): this is a further confirmation of the 'hard character' of bacterial classification.

A final remark has to do with the apparent 'triviality' of the problem, suggested by the reaching of 100% discrimination that could make one thinks that this is an 'easy task'. As first, it is important to stress that the problem is trivial only a-posteriori and only if the basic task is intended as a pure species-by-species discrimination, but this is not the focus of these experiments. There are much easier ways to perform such recognition, here the reaching of a 100% discrimination on the sole basis of the metabolic network wiring is a biological relevant result per se. Such result gives the biologist an unparalleled occasion to rely on a universal phenotype [253] (all organisms have a metabolism, only few have blood circulation or swimming activity) that puts together environmental and evolutionary dimensions. The granulation approach (at odds with other methods like K-NN) catches the essential property of metabolism to be organised into chain of chemical reactions that give rise to intermediate level organisation denominated 'pathways' that are biologically meaningful per se.

TABLE 5.32: Metabolic Networks classification, relevant edges for Problem 1 ($T = 50$). These edges have been selected (and survived the feature selection phase) in almost all of the stratified splits. $I^{(e)}$ and $I^{(p)}$ indicate the INDVAL scores for eukaryotes and prokaryotes, respectively. The leftmost column features "ℓ_1", "ℓ_2" or "$*$" if the edge has been selected by using ℓ_1-SVMs, ν-SVMs or has been selected by both classifier, respectively.

Classifier	Source Node	Target Node	$I^{(e)}$	$I^{(p)}$
$*$	cpd:C01194	cpd:C01277	99.7	0
$*$	cpd:C01212	cpd:C01050	0	89.5
$*$	cpd:C00133	cpd:C00993	0	90.3
ℓ_1	cpd:C01277	cpd:C04637	99	0
$*$	cpd:C03170	cpd:C02090	98.2	0
$*$	cpd:C15667	cpd:C04751	0	86.5
$*$	cpd:C01212	cpd:C00692	0	89.8
$*$	cpd:C06156	cpd:C00352	0	83.5
$*$	cpd:C04501	cpd:C06156	0	83
ℓ_2	gl:G00009	gl:G00171	86.9	0
$*$	cpd:C04598	cpd:C04317	93.6	0
$*$	cpd:C04631	cpd:C00043	0	89.9
ℓ_2	gl:G00143	gl:G00144	93.9	0
$*$	cpd:C01194	gl:G00143	97.9	0
ℓ_2	cpd:C00100	cpd:C02876	0	67.2
ℓ_1	cpd:C04421	cpd:C00666	0	73.4

5.5 Tests on Benchmark Datasets

So far, the techniques presented in Chapter 4 have been tested on real-world case studies, namely the study of proteins' shape-vs-function (Section 5.2), the study of proteins' shape-vs-solubility (Section 5.3) and the analysis of metabolic pathways

TABLE 5.33: Metabolic Networks classification, relevant edges for Problem 1 ($T = 50$). $I^{(an)}$, $I^{(ar)}$, $I^{(b)}$, $I^{(f)}$, $I^{(pl)}$ and $I^{(pr)}$ indicate the INDVAL scores for animals, archaea, bacteria, fungi, plants and protists, respectively.

Classifier	Source Node	Target Node	$I^{(an)}$	$I^{(ar)}$	$I^{(b)}$	$I^{(f)}$	$I^{(pl)}$	$I^{(pr)}$
$*$	cpd:C00522	cpd:C18911	0	93.1	0.002	0	0	0
$*$	cpd:C00036	cpd:C00149	0	0	0	0	97.06	0
$*$	cpd:C02139	cpd:C16540	0	0	0	0	97.06	0
$*$	cpd:C02280	cpd:C00096	0.007	0	0	0	94.01	0.51
$*$	cpd:C02280	cpd:C15926	0.005	0	0	0	93.52	0.16
$*$	cpd:C05440	cpd:C15778	0	0	0	93.01	0	0
$*$	cpd:C00681	cpd:C03372	0	0	0	68.89	0	0
$*$	cpd:C03715	cpd:C03968	0	0	0	68.89	0	0
$*$	cpd:C00157	cpd:C02737	98.49	0	0	0	0	0.01
$*$	cpd:C01245	cpd:C01272	98.15	0	0	0	0	0.15
$*$	cpd:C01277	cpd:C11554	97.02	0	0	0	0	0.18

TABLE 5.34: Metabolic Networks classification, relevant edges for Problem 3, class 2 only ($T = 50$). $I^{(b)}$, $I^{(f)}$, $I^{(i)}$, $I^{(m)}$ and $I^{(r)}$ indicate the INDVAL scores for birds, fishes, insects, mammals and reptiles, respectively.

Classifier	Source Node	Target Node	$I^{(b)}$	$I^{(f)}$	$I^{(i)}$	$I^{(m)}$	$I^{(r)}$
$*$	cpd:C00191	cpd:C05385	0	100	0	0	0

(Section 5.4). Nonetheless, the proposed pattern recognition techniques have to be intended as general purpose classification systems, not linked to a specific application per se. In order to investigate their capabilities, the three most cutting-edge techniques from Chapter 4 (the two embedding techniques and hypergraph kernels) have been tested against suitable competitors on well-known graph classification datasets (Section 5.1.4).

5.5.1 Hypergraph Kernels

The four proposed kernels, namely the Histogram Kernel (HK), the Weighted Jaccard Kernel (WJK), the Edit Kernel (EK) and the Stratified Edit Kernel (SEK) have been benchmarked against four state-of-the-art graph kernels, namely the Weisfeiler–Lehman Subtree Kernel (WL) [343], Weisfeiler–Lehman Shortest Path Kernel (WLSP) [343], Propagation Kernel (PK) [288] and Random Walk Kernel (RW) [366]. Since the underlying topology is available (i.e., for each graph within each dataset its adjacency matrix is known), the parameter-free Clique Complex is used (cf. Section 2.2) and evaluated using the Bron-Kerbosch algorithm [54, 65]. However, the parameter-free peculiarity is not true for the four competitors. Specifically, WL, WLSP and PK are iterative kernels, so one needs to find a suitable number of iterations, say h. Furthermore, PK needs the bin width, say w, for locality sensitive hashing and RW needs the walk decay factor λ.

Each dataset has been split in Training Set (70%) and Test Set (30%) and the kernel parameters, along with the regularisation term ν for ν-SVM, are learned via 5-fold cross-validation driven by random search [37] on the Training Set. The best parameters set is the one that maximises the cross-validation accuracy and it will finally be

used in order to evaluate the final performances on the Test Set[48]. Table 5.35 summarises the set of parameters to be tuned for the four proposed kernels and the three competitors. For PK, the following configuration has been used: the total variation distance has been used in order to evaluate hashes, label diffusion has been used for feature transformation (fully labelled graphs) and the base kernel has been set as the linear kernel.

TABLE 5.35: Parameters to be tuned (along with their respective admissible values) for each of the tested kernels. For the sake of completeness, the regularisation term ν for ν-SVMs has been included as well.

	Parameter			
	$\nu \in (0,1]$	$h \in \{1,\dots,20\}$	$w \in [10^{-8}, 10^{-1}]$	$\lambda \in (0,1)$
HK	✓	✗	✗	✗
WJK	✓	✗	✗	✗
EK	✓	✗	✗	✗
SEK	✓	✗	✗	✗
WL	✓	✓	✗	✗
WLSP	✓	✓	✗	✗
PK	✓	✓	✓	✗
RW	✓	✗	✗	✓

Since WL, WLSP, PK and RW have been implemented in MATLAB® by the respective Authors, the four proposed kernels have been implemented in MATLAB® as well in order to avoid platform differences. Results herein presented have been obtained with the following setup: a 64GB RAM workstation equipped with two 6-core hyper-threaded Intel® Xeon® CPU E5-2620 v3 @ 2.40GHz running MATLAB® R2018b on Linux Ubuntu 18.04.2 LTS. Multithreaded parallelism has been exploited in order to speedup the training phase, where each thread processes the training data by using a different set of parameters. For WLSP, the shortest path length matrices for all graphs in the dataset at hand have been evaluated beforehand, so that each thread is focused on the proper kernel matrix evaluation with the current parameter settings[49].

Figure 5.29 shows the accuracy on the test set obtained by the four competitors and the four proposed kernels averaged across five training-test stratified splits. For the sake of comparison, the same five splits have been fed to all tested kernels. From Figure 5.29 it is possible to see that the four proposed kernels (especially WJK, EK and SEK) in the vast majority of the cases outperform the three competitors. Furthermore, they are also very appealing for large datasets such as DD or the TOX21 family since no out-of-memory errors have been triggered and all executions terminated within the 24 hours deadline (for details, see caption of Figure 5.29).

[48]This experimental setup is quite different with respect to the current literature, so results may vary. Usually, the final results are obtained using a k-fold cross-validation (usually $k = 10$) on the entire dataset, without considering a separate Test Set. To the best of the author's judgement, considering a separate Test Set better highlights the generalisation capabilities of the learning system. Minor discrepancies with respect to related works include the utilisation of random search (instead of grid search) and the utilisation of ν-SVM instead of C-SVM [50, 86]. Acknowledging these discrepancies, results herein shown for WL, WLSP, PK and RW have been obtained from the ground-up thanks to the source code provided by the respective authors and are not taken from any research paper.

[49]The evaluation of the shortest paths matrices can be performed using the Floyd–Warshall algorithm [141, 372], which does not depend on any of the parameters from Table 5.35.

	HK	WJK	EK	SEK	WL	WLSP	PK	RW
AIDS	84.3	99.3	99.3	99.3	96.2	97.1	95.9	98.7*
BZR	78.7	83.3	84.4	84.3	78.7	78.7	78.7	83.6*
COX2	80.4	83.0	81.0	80.6	78.0	78.0	78.0	78.0*
DD	75.1	76.0	76.6	76.8	74.4	OoM	74.4*	OoM
DHFR	66.4	74.1	71.5	71.3	58.4	60.9	60.8	60.8*
ENZYMES	31.8	45.7	43.7	42.1	18.7	18.8	15.9	21.1
FIRSTMM_DB	43.1	40.0	46.1	38.5	24.6	OoM	27.7	23.1*
KKI	56.0	56.0	60.8	55.2	56.0	56.8	56.0	56.0*
MSRC_21	85.0	89.9	86.6	76.5	41.1	47.6	40.6	23.7*
MSRC_9	87.8	93.1	91.0	89.2	69.0	80.9	77.3	50.8
MUTAG	70.5	84.2	85.3	80.7	80.0	82.5	74.4	87.7
Mutagenicity	71.8	80.1	74.8	75.5	62.0	60.0*	61.0	56.1*
NCI1	62.1	74.4	67.2	67.8	59.6	57.8*	OoM	OoM
NCI109	58.2	74.4	67.7	68.4	62.9	56.8*	61.1	OoM
OHSU	53.3	53.3	53.3	54.2	54.2	54.2	54.2	54.2
PROTEINS	72.0	73.0	72.6	74.5	70.8	63.0*	62.4	74.5*
PTC_FM	60.8	59.6	60.2	60.4	59.0	59.4	59.0	59.0*
PTC_FR	66.0	64.5	64.0	64.3	63.0	65.1	65.1	65.1*
PTC_MM	64.8	63.4	61.8	64.0	61.4	61.4	61.4	61.4*
PTC_MR	57.5	60.2	54.2	56.1	55.8	56.0	52.3	55.8*
Peking_1	51.5	53.1	57.7	53.1	58.5	57.7	57.7	57.7*
SYNTHETIC	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0*
Tox21_AHR	88.4	89.8	88.4	88.4	88.4	OoM	OoM	OoM
Tox21_AR	95.9	96.8	96.0	96.0	OoM	OoM	OoM	OoM
Tox21_AR-LBD	96.4	97.6	96.5	96.5	OoM	OoM	OoM	OoM
Tox21_ARE	84.5	85.9	84.7	84.7	84.7	83.8*	OoM	OoM
Tox21_ATAD5	96.3	96.6	96.3	96.3	OoM	OoM	OoM	OoM
Tox21_ER	87.7	88.5	87.8	87.8	OoM	OoM	OoM	OoM
Tox21_ER_LBD	94.9	95.5	94.9	94.9	OoM	OoM	OoM	OoM
Tox21_HSE	94.9	95.5	94.8	94.8	OoM	OoM	OoM	OoM
Tox21_MMP	82.9	87.9	84.7	84.6	84.1	OoM	OoM	OoM
Tox21_PPAR-γ	96.8	97.4	97.3	97.3	OoM	OoM	OoM	OoM
Tox21_aromatase	95.0	95.7	95.0	95.0	OoM	OoM	OoM	OoM
Tox21_p53	93.8	94.6	93.8	93.8	OoM	OoM	OoM	OoM

FIGURE 5.29: (Hyper)graph kernels comparison, average accuracy on the test set. The colour scale has been normalised row-wise (i.e., for each dataset) from yellow (lower values) towards green (higher values, preferred). 'OoM' indicates that the training procedure went out-of-memory. The asterisk indicates that the training procedure for at least one of the 5 training-test splits did not finish within 24 hours: in this case, the training procedure has been aborted and the best parameters obtained so far have been retained. For the sake of readability, standard deviations are not included.

Conversely, Figure 5.30 shows the running times (in seconds) for evaluating the kernel matrix over the entire dataset (no training-test split). Wall-clock times refer to a single-threaded evaluation, no parallelism has been exploited. Since the execution is not deterministic due to spurious processes running in background, the running times have been averaged across five evaluations of the kernel matrices. For WL, WLSP, PK and RW, the best parameters across the aforementioned five training-test splits have been used. From Figure 5.30 it is possible to see RW is by far the slowest kernels to compute, followed by EK and SEK. The latter two kernels lack any vectorised statements, whereas the same is not true for HK and WJK whose computational burden is competitive with WL and PK (generally, the fastest kernel). Notwithstanding the running times as such, it is worth stressing that WL, WLSP, PK and RW shall be evaluated several times in order to find a suitable set of parameters (see Table 5.35), whereas HK, WJK, EK and SEK can be evaluated only once.

	HK	WJK	EK	SEK	WL	WLSP	PK	RW
AIDS	3.9	5.2	139.4	290.8	4.1	5.3	0.9	79163.4
BZR	1.2	1.2	14.1	18.8	2.2	2.8	0.2	872.8
COX2	1.5	1.6	21.0	27.2	15.6	38.4	0.5	271.3
DD	88.1	217.9	1447.3	1221.9	45.8	OoM	88.5	OoM
DHFR	2.4	2.5	52.7	70.6	8.9	13.9	0.9	928.2
ENZYMES	1.9	1.9	27.6	47.1	4.3	2.9	0.4	155.9
FIRSTMM_DB	13.9	13.7	56.2	36.2	8.0	OoM	2.7	3714.5
KKI	0.3	0.4	0.7	1.1	1.2	1.0	0.1	2893.2
MSRC_21	5.4	6.0	92.1	106.4	5.3	141.1	1.8	9714.4
MSRC_9	1.1	1.1	7.3	9.1	2.0	1.6	0.1	116.5
MUTAG	0.4	0.4	2.0	2.9	0.5	0.6	0.1	39.2
Mutagenicity	11.1	13.7	1146.9	1855.5	37.7	100.2	20.6	117058.2
NCI1	11.0	14.7	1078.1	1718.3	36.9	264.3	OoM	OoM
NCI109	11.1	15.1	1076.8	1735.3	43.1	261.5	19.4	OoM
OHSU	1.4	1.5	2.7	2.6	2.7	2.5	0.3	10051.1
PROTEINS	4.1	4.4	108.9	177.5	26.2	91.4	2.7	783.6
PTC_FM	0.6	0.6	4.4	8.2	1.5	3.0	0.1	644.5
PTC_FR	0.6	0.7	4.6	8.5	3.1	3.6	0.1	669.9
PTC_MM	0.6	0.6	4.1	7.6	4.0	1.2	0.1	650.0
PTC_MR	0.6	0.6	4.3	8.2	1.2	1.6	0.1	536.9
Peking_1	0.5	0.6	1.1	1.6	0.4	2.7	0.0	3957.7
SYNTHETIC	3.0	3.1	34.7	34.8	5.8	18.4	0.9	736.7
Tox21_AHR	16.6	41.1	2539.3	5088.3	66.2	OoM	OoM	OoM
Tox21_AR	17.3	45.5	3340.5	6765.4	OoM	OoM	OoM	OoM
Tox21_AR-LBD	19.5	55.5	2743.3	5549.2	OoM	OoM	OoM	OoM
Tox21_ARE	13.6	30.8	1789.7	3737.6	50.0	53.7	OoM	OoM
Tox21_ATAD5	18.4	47.9	3147.4	6366.2	OoM	OoM	OoM	OoM
Tox21_ER	15.2	35.2	2222.2	4510.3	OoM	OoM	OoM	OoM
Tox21_ER_LBD	17.9	46.1	2952.3	6005.9	OoM	OoM	OoM	OoM
Tox21_HSE	15.9	39.2	2384.9	4929.1	OoM	OoM	OoM	OoM
Tox21_MMP	14.5	33.3	1989.8	4067.3	79.8	OoM	OoM	OoM
Tox21_PPAR-γ	16.4	42.0	2473.3	5080.6	OoM	OoM	OoM	OoM
Tox21_aromatase	14.5	31.9	1948.3	3969.2	OoM	OoM	OoM	OoM
Tox21_p53	17.8	47.1	2951.5	5958.8	OoM	OoM	OoM	OoM

FIGURE 5.30: (Hyper)graph kernels comparison, average running times (in seconds) for evaluating the kernel matrix on the entire dataset. The colour scale has been normalised row-wise (i.e., for each dataset) from yellow (lower values, preferred) towards green (higher values). 'OoM' indicates that the training procedure went out-of-memory, hence the impossibility to gather a suitable set of parameters. For the four proposed kernels, running times also include the simplicial complexes evaluation starting from the underlying graphs. For the sake of readability, standard deviations are not included.

In Section 4.5 it has been said that, whilst HK and WJK satisfy Mercer's condition, the same might not be true for edit-based kernels (namely, EK and SEK). In order to quantify the goodness of the two proposed edit-based kernels, the negative eigenfraction (NEF) is investigated, defined as the relative mass of the negative eigenvalues [311]:

$$NEF = \frac{\sum_{i:\lambda_i<0} |\lambda_i|}{\sum_i |\lambda_i|}. \tag{5.20}$$

Clearly, $NEF \in [0,1]$: the closer to 0, the better. Figure 5.31 shows the NEF of the kernel matrices evaluated over the 34 datasets from Table 5.2 for EK and SEK. Both the proposed kernels have rather low NEF, the maximum being 0.0605 (dataset TOX21_ATAD5 when using EK). Another interesting aspect is that SEK always outperforms EK in terms of NEF for all datasets (i.e., always less than or equal to).

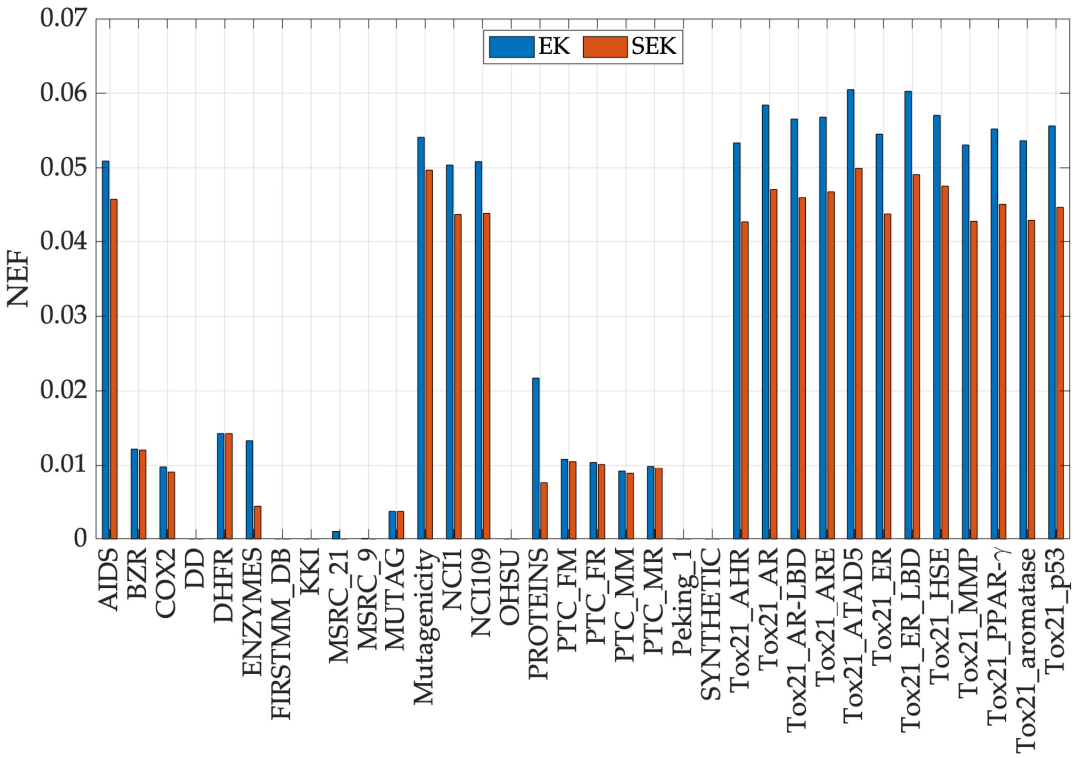


FIGURE 5.31: Edit-based hypergraph kernels, negative eigenfraction.

A final facet to be stressed, lies on the graphs-vs-hypergraphs duality: indeed, the datasets used for experiments are graph datasets (not hypergraphs), yet the proposed kernels are hypergraph kernels. The possibility to infer the simplicial structure from the underlying graph (e.g., thanks to the flag complex) has a crucial role in this regard, thanks to which one can infer hypergraphs starting from graphs [27, 28, 151]. This makes the four proposed kernels comparable with graph kernels (i.e., they both start from the same graphs). Notwithstanding that, it is safe to say that the four proposed kernels are indeed hypergraph kernels since they work on the top of the simplicial complexes obtained from the underlying 1-skeletons.

5.5.2 Embedding over Simplicial Complexes

For a thorough investigation of the proposed embedding procedure, both of the classification strategies (ν-SVM and ℓ_1-SVM) have been considered. Two classification systems have been used as competitors:

Weighted Jaccard Kernel Already known from Section 4.5 and whose classification capabilities have been shown in Section 5.5.1

GRALG Originally proposed in [41] and later used in [43] and [44] for image classification, GRALG is a GrC-based classification system for graphs. Despite it considers network motifs rather than simplices, it is still based on the same embedding procedure via symbolic histograms. In small terms, GRALG extracts network motifs from the training data and runs a clustering procedure on such subgraphs by using a graph edit distance as the core (dis)similarity measure. The medoids (MinSODs) of these cluster form the alphabet on the top of which the embedding space is built. Two genetic algorithms take care of tuning the alphabet synthesis and the feature selection procedure, respectively. GRALG, however, suffers from an heavy computational burden which

may become unfeasible for large datasets. In order to overcome this problem, the random walk-based variant proposed in [17], and later explored in [16, 18, 19], has been used.

The rationale behind these competitors follows: the WJK relies as well on simplicial complexes, despite it performs an implicit embedding; conversely, GRALG is another GrC-based classification system powered by symbolic histograms, despite it does not rely on simplicial complexes.

Thirty datasets drawn from Table 5.2 have been considered for testing. Each dataset has been split in training set (70%) and test set (30%) in a stratified manner in order to preserve ground-truth labels distribution across the two splits. Validation data has been taken from the training set via 5-fold cross-validation. For the proposed embedding procedure via simplicial complexes and WJK, the Clique Complex has been used since the underlying 1-skeleton is already available from the considered datasets. For GRALG, the maximum motifs size has been set to 5 and, following [17], a subsampling rate of 50% has been performed on the training set[50]. Alongside GRALG and WJK, the accuracy of the dummy classifier is also included [113]: the latter serves as a baseline solution and quantifies the performance obtained by a purely random decision rule. Indeed, the dummy classifier outputs a given label, say l_i with a probability related to the relative frequency of l_i amongst the training patterns and, by definition, does not consider the information carried out by the pattern descriptions (input domain) in training data.

In Figure 5.32 the accuracy on the Test Set is shown for the five competitors: the dummy classifier, WJK, GRALG and the proposed embedding procedure using both non-linear ν-SVM and ℓ_1-SVM. In order to take into account intrinsic randomness in stratified sampling and in genetic optimisation, results herein presented have been averaged across five different runs. Clearly, for the tested datasets, a linear classifier performs poorly: it is indeed well-known that for high-dimensional datasets non-linear and linear methods may have comparable performances [88, 135]. As a matter of fact, for these datasets, PEKING_1 leaded to the largest embedding space (approx. 1500 symbols), followed by MSRC_9 (approx. 220 symbols).

From Figures 5.32 emerges that WJK is generally the most performing method, followed by the proposed embedding procedure, in turn followed by GRALG. Indeed, WJK exploits the entire simplicial complexes, without 'discarding' any simplices due to the explicit (and optimised) embedding procedure, as proposed in this work. Further, the evaluation of the kernel matrix sees a 1-vs-1 comparison, so only simplices belonging to the two simplicial complexes are subject to matching, without relying on simplices belonging to other patterns. Amongst the three methods, WJK is also the fastest to train: the kernel matrix can be pre-evaluated using very fast vectorised statements and the only hyperparameter that needs to be tuned is the ν-SVM regularisation term which, due to the boundedness, can be performed by a plain random search in $(0, 1]$. Amongst the two information granulation-based techniques, the proposed system outperforms GRALG in the vast majority of the cases. This not only has to be imputed to the modelling capabilities offered by hypergraphs, but also has a merely computational facet: the number of simple paths is much greater than the number of simplices[51], hence GRALG needs a 'compression stage' (i.e., clustering procedure via BSAS) to return a feasible number of alphabet

[50] Four datasets from the original 34 shown in Table 5.2 have been discarded because of their heavy connectivity: a 50% subsampling rate was not sufficient for GRALG to avoid out-of-memory errors.

[51] A graph with n vertices has $\mathcal{O}(n!)$ paths, whereas the number of cliques goes like $\mathcal{O}(3^{n/3})$.

symbols. This compression stage not only may impact the quality of the embedding procedure, but also leads to training times incredibly higher with respect to the proposed technique in which simplices can be interpreted as granules themselves.

Another interesting aspect that should be considered for comparison relies on the model interpretability. Despite WJK seems the most appealing technique due to high training efficiency and remarkable generalisation capabilities, it basically relies on pairwise evaluations of a positive definite kernel function between pair of simplicial complexes which can then be fed into a kernelised classifier. This *modus operandi* does not make the model interpretable and no knowledge discovery phase can be pursued afterwards. The same is not true for GrC-based pattern recognition systems such as GRALG or the proposed one, as already addressed in the EC classification case study (Section 5.2.6).

	WJK	Embedding + ν-SVM	Embedding + $L1$-SVM	GRALG	Dummy
AIDS	99.5	99.3	66.9	98.3	68.0
BZR	84.5	83.8	64.5	83.5	66.6
COX2	82.4	80.7	79.9	77.7	65.9
DHFR	74.9	72.7	65.8	69.7	52.4
ENZYMES	46.0	43.1	28.8	30.3	16.6
KKI	60.0	55.8	36.7	45.8	50.6
MSRC_9	92.7	89.7	90.9	87.6	12.8
MUTAG	84.6	84.6	76.4	78.9	55.5
Mutagenicity	79.3	77.0	72.2	71.7	50.6
NCI1	74.6	72.7	62.7	71.2	50.0
NCI109	73.8	71.4	58.7	72.7	50.0
PROTEINS	72.4	71.3	66.1	70.7	51.8
PTC_FM	60.0	60.0	56.1	61.9	51.6
PTC_FR	65.5	65.0	63.0	62.5	54.8
PTC_MM	64.8	63.4	60.2	60.0	52.7
PTC_MR	60.0	58.2	55.3	58.8	50.7
Peking_1	56.0	65.6	52.8	56.0	51.2
SYNTHETIC	50.0	50.0	50.0	50.0	50.2
Tox21_AHR	90.0	89.3	52.8	88.1	79.5
Tox21_AR	97.0	96.8	74.5	96.4	92.2
Tox21_AR-LBD	97.6	97.3	80.3	96.8	93.2
Tox21_ARE	85.9	85.2	54.7	84.1	74.0
Tox21_ATAD5	96.6	96.5	39.5	96.2	92.8
Tox21_ER	88.4	88.6	64.8	88.0	78.6
Tox21_ER_LBD	95.5	95.4	72.1	94.9	90.3
Tox21_HSE	95.5	95.1	66.2	94.8	90.0
Tox21_MMP	87.5	86.5	55.4	84.8	73.7
Tox21_PPAR-γ	97.5	97.4	50.1	97.3	94.7
Tox21_aromatase	95.6	95.6	49.5	94.9	90.5
Tox21_p53	94.6	94.2	68.4	93.4	88.3

FIGURE 5.32: Embedding over simplicial complexes performances on benchmark data, average accuracy on the Test Set. The colour scale has been normalised row-wise (i.e., for each dataset) from yellow (lower values) towards green (higher values, preferred). For the sake of readability, standard deviations are not included.

5.5.3 Embedding via INDVAL

Twenty-two datasets from Table 5.2 have been considered for this preliminary experiments in which the INDVAL score acts as the main role in information granule synthesis. Furthermore, in order to prove the generalisation of the procedure, let us define a k-cutoff path as a simple path composed by k edges ($k+1$ nodes).

The INDVAL score can be generalised as follows, for a given value k

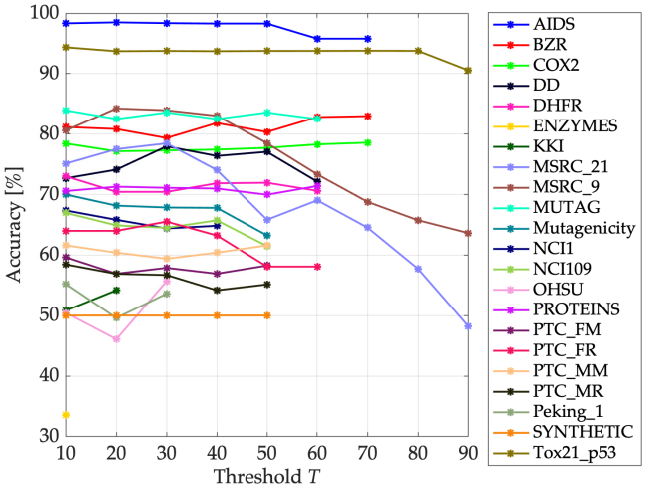
$$A_{i,j} = \frac{\text{\# patterns in class } j \text{ having } k\text{-cutoff path } i}{\text{\# patterns having } k\text{-cutoff path } i} \tag{5.21}$$

$$B_{i,j} = \frac{\text{\# patterns in class } j \text{ having } k\text{-cutoff path } i}{\text{\# patterns belonging to class } j} \tag{5.22}$$

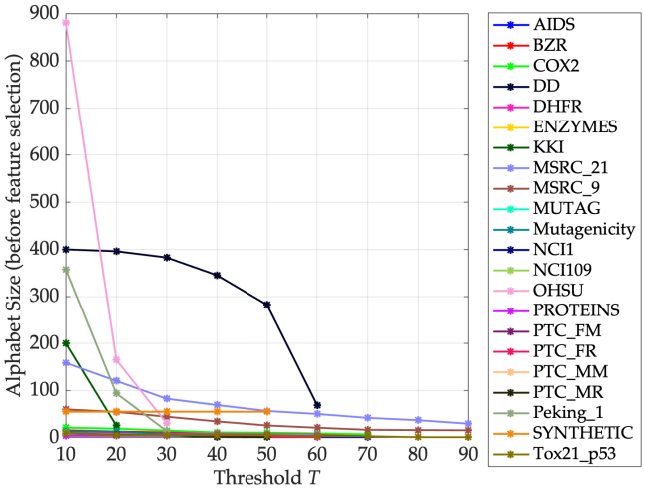
$$I_{i,j} = A_{i,j} \cdot B_{i,j} \cdot 100 \tag{5.23}$$

Trivially, by considering that an edge is a k-cutoff path with $k = 1$, then Eqs. (5.21)–(5.23) collapse into Eq. (4.16)–(4.18).

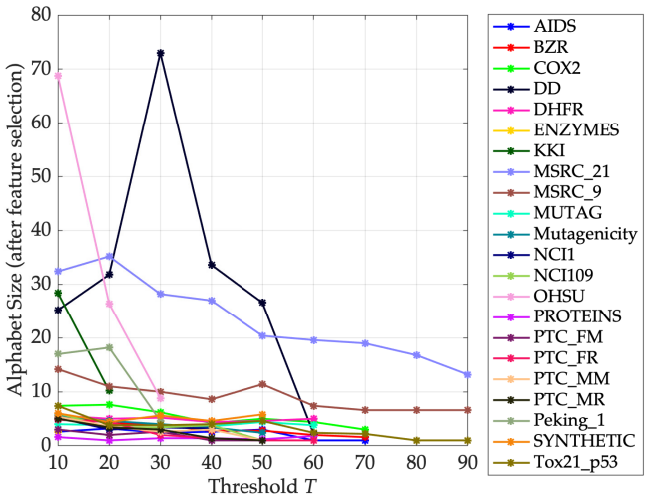
Figures 5.32, 5.32 and 5.32 show the accuracy on the Test Set for the twenty-two considered datasets, along with the initial alphabet size (after threshold T) and the final alphabet size (after genetic optimisation). Results are averaged across five Training-Validation-Test Set splits. Interrupted line arts mean that the alphabet is empty for that T value; that is, there are no symbols with INDVAL above the threshold and the embedding cannot be performed.



(A) Average accuracy on the Test Set

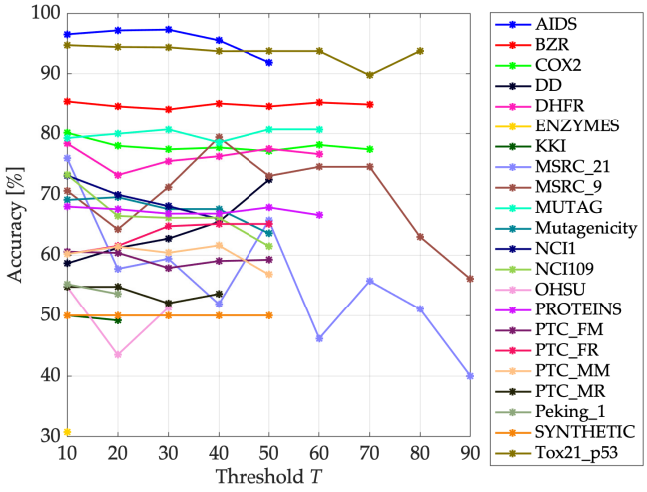


(B) Prior symbols selection

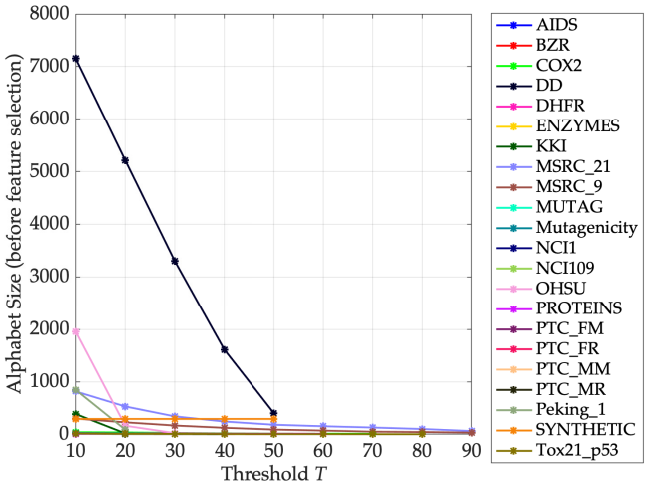


(C) Optimised symbols selection

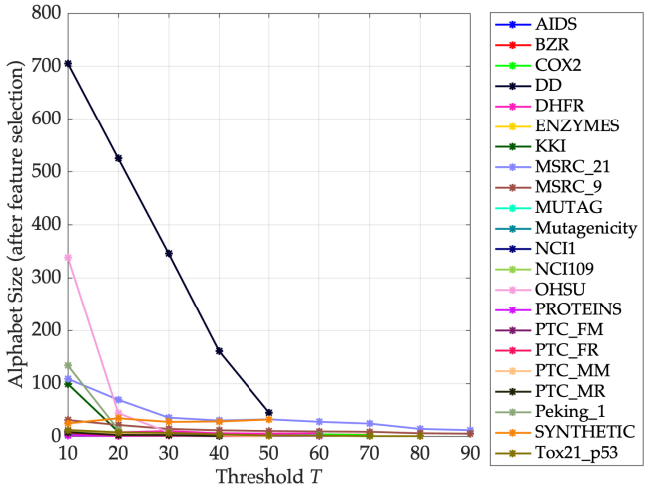
FIGURE 5.32: Embedding via INDVAL (path length 1) performances on benchmark data.



(D) Average accuracy on the Test Set

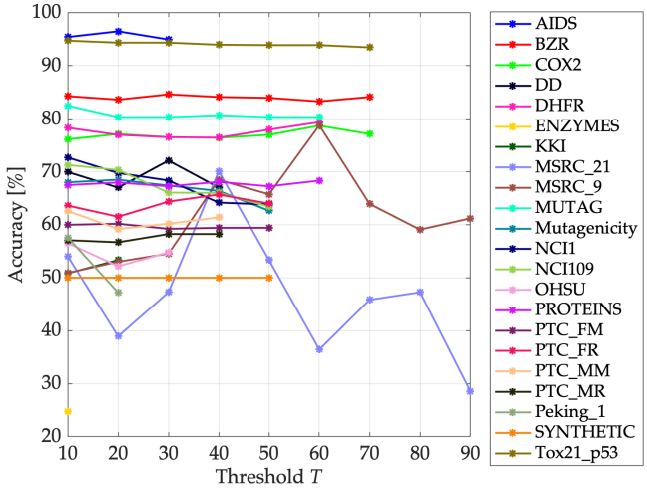


(E) Prior symbols selection

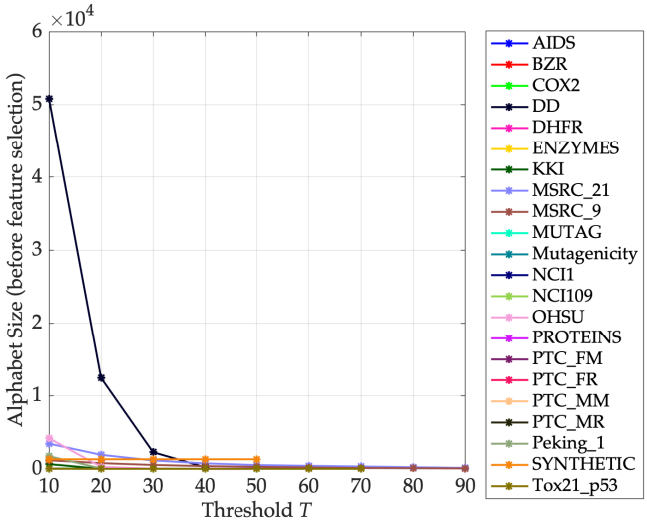


(F) Optimised symbols selection

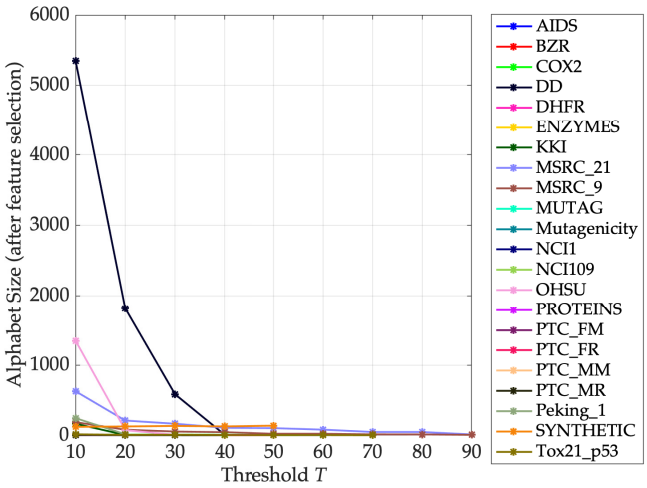
FIGURE 5.32: Embedding via INDVAL (path length 2) performances on benchmark data.



(G) Average accuracy on the Test Set



(H) Prior symbols selection



(I) Optimised symbols selection

FIGURE 5.32: Embedding via INDVAL (path length 3) performances on benchmark data.

Chapter 6

Other Research Activities

6.1 Evolutive Agent-Based Clustering

Evolutive Agent Based Clustering (E-ABC), originally proposed in [259], is a multi agent-based clustering algorithm whose roots trace back to a previous work from the research groups: Local Dissimilarities-Agent-Based Clusters Discoverer (LD-ABCD) [42]. LD-ABCD is a multi agent algorithm in which agents perform a Markovian random walk on a weighted graph representation of the input dataset. Specifically, each agent builds its own graph connection matrix amongst data points, weighting the edges according to its own set of distance measure parameters and performs a random walk on such graph in order to discover clusters. Then, each agent autonomously decides whether the processed data (i.e., the path of the walk) can be accepted as a valid cluster or not. This algorithm has been successfully employed in [45] to identify frequent behaviours of mobile network subscribers starting from a set of call data records.

LD-ABCD allows the introduction of *metric learning* in pattern recognition [368], namely the ability of an intelligent system to learn suitable parameters for a (parametric) dissimilarity measure in order to optimise some predefined criteria. Metric learning tasks can be divided in two main family:

Global metric learning: learning a set of parameters for the dissimilarity measure suitable for solving the entire pattern recognition problem. To this family usually belongs the feature selection paradigm (see Section 3.2), hence the vast majority of the case studies presented in Chapter 5. Indeed, in those tests, a feature selector vector (usually referred to as $\mathbf{w}$) used to discard unpromising features in order to maximise the classification performances.

Local metric learning: learning a set of parameters for the dissimilarity measure in a cluster-wise or class-wise fashion. For the sake of ease, let us consider a clustering problem in a Euclidean space: there might exist problems in which clusters lie in different subspaces, so the learning system not only must find clusters (as usual also in 'standard' clustering algorithms), but also the subspace in which each cluster lies (i.e., is well-formed).

LD-ABCD definitely falls under the local metric learning umbrella (i.e., each agent build its own graph-based representation of the dataset according to its own set of dissimilarity measure parameters), yet it is featured by a huge computational complexity, which drastically explodes as the dataset size increases because each agent operates on (a graph-based representation of) the entire dataset.

E-ABC aims at working in a dual fashion, where many agents operate on a small randomly drawn subset of the available patterns. In E-ABC, each agent is in charge of running a core, lightweight clustering algorithm on a randomly chosen subset $\overline{\mathcal{D}}$

of the entire dataset $\mathcal{D}$. Each agents runs a variant of the BSAS algorithm (namely, RL-BSAS [324]) which adds some Reinforcement Learning-based behaviour to the standard BSAS by means of two additional parameters (reward factor α and forgetting factor β), alongside the well-known cluster radius θ. E-ABC must be able to discover clusters in specific subspaces. To this end, the dissimilarity measure core of the clustering algorithm reads as a weighted Euclidean distance (cf. Eq. (5.2)) with binary weights $\mathbf{w}$. The clustering algorithm parameters and the weights vector are optimised by means of a genetic algorithm which aims at maximising compactness and cardinality in order to foster the swarm of agents towards well-formed clusters. Before moving to the next generation, agents undergo an intra-agent fusion phase and an inter-agent fusion phase in order to shrink the output size, namely the set of clusters found by the swarm.

In order to test the E-ABC capabilities of finding well-formed cluster lying in different subspaces, eight synthetic datasets have been considered composed by a juxtaposition of informative features (drawn from a multivariate Gaussian distribution) and noisy features (drawn from a uniform distribution). Those datasets differ on the number of noisy features and/or the number of patterns per cluster. These preliminary exercises showed that (at least for the considered datasets with strong assumption on Gaussian clusters distribution and with tight and separate clusters) E-ABC is capable of estimating quite correctly the most important characteristics of the clusters and paved the way for further ideas which include the possibility of dealing with structured data by properly replacing centroids with medoids in BSAS and by moving away from the Euclidean distance, the possibility of developing a supervised version for classification purposes, the adoption of customised evolutionary strategies for dealing with multimodal problems (see e.g. [223]) for better capturing different subspaces and, obviously, the possibility of letting the agents to work in parallel by exploiting multi-core and/or many-core architectures due to the lightweight clustering procedures which can be performed independently.

6.2 Distributed k-medoids Clustering

As briefly sketched in Section 5.4 in the context of clustering metabolic networks belonging to organisms known to populate the human intestinal tract, k-medoids is a hard partitional clustering algorithm and, as such, it breaks the dataset into k non-overlapping clusters in such a way that the WCSoD (Eq. (5.15)) is minimised. In literature, there are several algorithms for solving the k-medoids problem, including PAM [191, 192], CLARA [191, 192], CLARANS [291] and Voronoi iteration [301]. Regardless of the algorithm, k-medoids is harder to solve with respect to the widely-known k-means due to the definition of the medoid itself: whilst the centroid (k-means) is defined as the centre of mass of the cluster, the medoid is defined as the element of the cluster which minimises the sum of pairwise distances. As the Voronoi iteration method is concerned, naïvely speaking, for a given cluster C, one needs to evaluate a $|C| \times |C|$ matrix encoding the pairwise dissimilarities between all items in C and then find the element which minimises the sum of distances. This adds a quadratic complexity procedure, whereas in k-means the centroid can be evaluated in linear time with respect to $|C|$.

In the big data era, the need for processing massive datasets emerged: to this end, efficient and rather easy-to-use large-scale processing frameworks such as MapReduce [103] and Apache Spark [400] have been proposed, gaining a lot of attention from computer scientists and machine learning researchers alike. As a matter of fact,

several machine learning algorithms have been grouped in MLlib [270], the machine learning library built-in in Apache Spark.

In short, MapReduce is a parallel and distributed workflow which consists in two phases: a *Map* phase and a *Reduce* phase. During the Map phase, a map function is applied to a dataset shard and some intermediate results are returned. Those results are usually presented in a ⟨key, value⟩ pair form. The MapReduce framework automatically groups values according to their keys in such a way that each key k is associated to the list of its values $[v_1, \ldots, v_n]$. The Reduce phase, finally, applies an aggregation function by properly combining values associated to the same key in order to return the final results. Since usually the Map phase returns a huge number of ⟨key, value⟩ records, MapReduce can suffer from high overhead and network congestion: in order to overcome this problem, a *Combiner* phase can be placed in-between Map and Reduce where results coming from a single Map worker are combined (i.e., 'reduced') before sending the results to the proper Reduce phase which aims at reducing only values belonging to the same key, but coming from different workers. Figure 6.1 shows the schematic MapReduce workflow for the word-count problem.

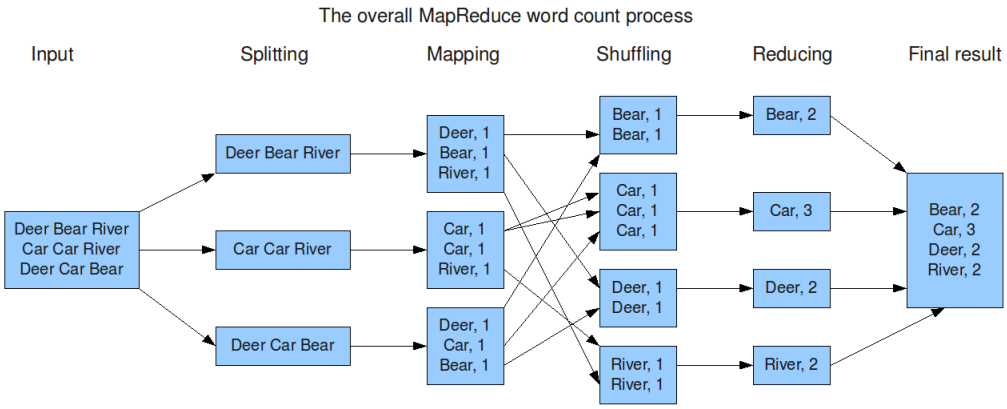


FIGURE 6.1: MapReduce schema for the word-count problem. The workflow does not include a Combiner phase: if one has to include this optional phase, for example, the second mapper shall return ⟨Car, 2⟩ instead of two records of the form ⟨Car, 1⟩. Credits: Dr. Simone Scardapane, lecture notes.

MapReduce certainly is simple: the developer must only code the Map function, the Reduce function and, eventually, the Combiner function, whereas the framework automatically takes care of the rest (data splitting and forwarding to the mappers, shuffling and forwarding to reducers, and so on), yet the developer is forced to cast the algorithm into this very tight and rigorous workflow. Furthermore, MapReduce suffers from the following two drawbacks regarding

- iterative algorithms: MapReduce is doomed to be inefficient as computing nodes (both mappers and reducers) have no memory of past executions. In other words, at each iteration, data must be forwarded to computing nodes again.
- operations such as filtering, join and the like must be cast into Map and Reduce phases.

Whilst keeping the simplicity of MapReduce, Apache Spark emerged as a breakthrough framework for distributed processing. Spark overcomes these problems as it includes highly efficient `map`, `reduce`, `join`, `filter` (and many others) operations which are not only natively distributed, but can be arbitrarily pipelined together. As far as iterative algorithms are concerned, Spark allows caching in memory and/or disk, therefore there is no need to forward data back and forth from/to workers at each iteration.

Distributed implementations of the k-means algorithm have been widely discussed in literature (see e.g. [408] for a MapReduce implementation and [270] for an Apache Spark implementation) all of which basically leverage on the following observations:

- the point-to-centroid assignments can be computed independently in parallel
- the mean value relies on summing items together and the sum is an associative and commutative operation, so centroids can be updated efficiently.

So it is worth investigating whether the k-medoids using the Voronoi method can be parallelised in Apache Spark. This investigation has roots in the following rationales:

- the Voronoi method is at the basis of k-means as well, so one can count on an affirmed parallelisation workflow
- Apache Spark is way more efficient than MapReduce, especially for iterative algorithms.

Originally proposed in [256], a first implementation deals with real-valued patterns in order to prove both effectiveness against well-known implementations of the k-medoids algorithm and its efficiency in distributed scenarios. The first implementation features, as in k-means, a distributed point-to-medoid evaluation and then, one cluster at the time, the representative update takes place. In this regard, two alternatives have been proposed:

1. an exact medoid update routine: which solves, in a distributed manner, the entire quadratic problem underneath the medoid evaluation
2. an approximate medoid tracking procedure, based on [107]: a pool of size P hosts the first P patterns belonging to the cluster and then, by means of a stochastic random sampling, two competing patterns are matched against the current medoid and the farthest one is removed and replaced by one of the remaining patterns in the cluster.

In order to show the effectiveness of the distributed implementation, namely its ability to converge to established (even serial) implementations of the k-medoids algorithm, two well-known datasets from the UCI Machine Learning Repository [121, 224] (IRIS and WINE) have been considered. Starting from the very same set of initial medoids, the distributed implementation using the exact medoid update routine is able to converge to the same solution (i.e., WCSoD) of the MATLAB® R2015a k-medoids implementation. Due to its stochastic nature, the same is not true for the approximate medoid tracking procedure, yet by properly choosing a suitable pool size P the gap with respect to the exact solution can be reduced.

In order to show the efficiency of the distributed implementation, six additional datasets have been downloaded from the UCI Machine Learning Repository. Speedup

and sizeup have been used as suitable performance indices in order to address how a distributed algorithm is able to deal as the processing power increases whilst keeping the dataset size constant (the former) and how it is able to deal as the processing power remains constant, but the dataset size increases (the latter).

From the sizeup analysis emerged that the approximate medoid tracking routine has very good sizeup performances (e.g., a 4-times larger dataset needs from 1.9 to 5.5 times for time). The exact medoid update generally has lower performances, as expectable, yet as the dataset size increases the performances tend to improve.

From the speedup analysis emerged that the approximate medoid tracking is still the most appealing routine. The exact update method is concerned has very good speedup performances as well: indeed, due to the intensive parallel evaluations required for building the exact dissimilarity matrix, more and more workers are crucial for speeding up this computationally expensive phase.

A second implementation, proposed in [257], adds further improvements to the previous work, namely:

- improved dissimilarity matrices evaluation in case of small clusters
- improved dissimilarity matrices evaluation for the items in the pool in case of approximate medoid tracking
- new routine for initialing clusters that loose all of its members.

This work also features a new test campaign which involves structured data rather than real-valued vectors:

1. four metabolic pathways datasets have been downloaded from KEGG: the reference pathway itself along with three additional pathways (Biosynthesis of antibiotics, Biosynthesis of secondary metabolites, Microbial metabolism in diverse environments). The network representation of these pathways have been considered by relying on the maximal set of intervening metabolites: as in Section 5.4, networks can easily be compared by means of the Hamming distance
2. four primary structures datasets have been downloaded from UniProt for the following organisms: Homo sapiens, Escherichia coli str. K12, Drosophila melanogaster and Saccharomyces cerevisiae. Since primary structures are sequences (of amino-acids), the Levenshtein distance has been used as suitable dissimilarity measure.

From the speedup analysis emerged that exact medoid update shows a very good speedup performances, which approaches the linear behaviour (desired) as the dataset size increases, meaning that large datasets can be treated efficiently, yet –as already observed from the vector data tests– it suffers from low sizeup performances (e.g., a 4-times larger datasets needs around 10 times more time). The approximate medoid tracking behaves in a dual fashion with respect to the exact medoid update: whilst the latter outperforms the former in terms of speedup, the former outperforms the latter in terms of sizeup.

From the sizeup analysis emerged that the approximate medoid tracking greatly outperforms the exact medoid update, showing a remarkable sub-linear behaviour (e.g., a 4-times larger dataset needs approximately 1.3 times more time).

Results both in terms of sizeup and speedup are coherent between the two different scenarios (metabolic networks and primary sequences), suggesting that the overall implementation is quite robust with respect to the dissimilarity measure adopted.

To this point one might ask how to validate large-scale clustering results, for example, by means of internal validation indices such as

Silhouette index: The Silhouette index [331] is an internal clustering validation index which measures how-well a given point has been assigned to its own cluster. Let $\overline{d}_j^{(i)}$ indicate the average distance between a generic data point $\mathbf{x}_i$ and all patterns belonging to cluster j. For point $\mathbf{x}_i$ it is possible to define $a(\mathbf{x}_i)$ as the average dissimilarity from all the patterns within its own cluster, say $\mathcal{S}_h$, thus $a(\mathbf{x}_i) = \overline{d}_h^{(i)}$. Further, it is possible to define $b(\mathbf{x}_i)$ as the nearest-cluster average dissimilarity of which $\mathbf{x}_i$ is not a member, thus $b(\mathbf{x}_i) = \min_{j=1,\ldots,k}(\overline{d}_j^{(i)})$, with $j \neq h$. The Silhouette for $\mathbf{x}_i$ is evaluated as:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{b(\mathbf{x}_i), a(\mathbf{x}_i)\}} \tag{6.1}$$

and the overall Silhouette (i.e. for the whole clustering solution) is taken by averaging each data point's score:

$$s = \frac{1}{N}\sum_{i=1}^{N} s(\mathbf{x}_i) \tag{6.2}$$

By definition, $s(\mathbf{x}_i) \in [-1, +1]$ $(\forall i = 1, \ldots, N)$ and, by extension, $s \in [-1, +1]$ as well: the closer to $+1$, the better the clustering solution.

Davies-Bouldin index: The Davies-Bouldin index [98] is an internal clustering validation index which measures the intra-cluster separation against the inter-cluster variance. Let S_i be the statistical dispersion of the i^{th} cluster, computed as the average distance between the patterns belonging to the cluster itself and its representative (i.e. its medoid). Let $M_{i,j}$ be the distance between i^{th} and j^{th} clusters' representatives. For a clustering solution to be good, S_i should be as small as possible, whereas $M_{i,j}$ should be as large as possible; therefore for a given pair of clusters, a penalty score can be defined as:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{6.3}$$

The Davies-Bouldin Index for the i^{th} cluster is thus defined as

$$DBI_i = \max_{j \neq i} R_{i,j} \tag{6.4}$$

and the overall Davies-Bouldin Index (i.e. for the whole clustering solution) is taken by averaging each cluster's score:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} DBI_i \tag{6.5}$$

Conversely to the Silhouette, the Davies-Bouldin index is not bounded within a given range. As a general rule, the lower the Davies-Bouldin index, the better the clustering solution. The soundness of this result is a direct consequence of what previously explained; indeed, in order to obtain a low Davies-Bouldin index, the numerator in Eq. (6.3) should be small (i.e. compact clusters) and/or the denominator should be large (i.e. well separated clusters, far apart from each other).

Simplified Silhouette index: The Simplified Silhouette index [89, 126, 173, 364, 369] overcomes the problem of the Silhouette index as the latter needs to evaluate all pairwise distances between points. In the Simplified Silhouette Index, $a(\mathbf{x}_i)$ is defined as the distance between $\mathbf{x}_i$ and its cluster's representative, whereas $b(\mathbf{x}_i)$ is defined as the minimum distance with respect to the other clusters' representatives. $s(\mathbf{x}_i)$ is still evaluated using Eq. (6.1) and, by extension, the overall Simplified Silhouette Index is evaluated using Eq. (6.2). Like its 'standard' counterpart, the Simplified Silhouette Index is also in range $[-1, +1]$: the closer to $+1$, the better the clustering solution.

In [255] two contributions have been proposed:

1. the possibility of exploiting the full pairwise distance matrix in order to speedup the entire k-medoids evaluation, along with the evaluation of the Davies-Bouldin Index and the Silhouette Index (in fact, all of them rely only on dissimilarities between patterns), but this approach is trivially not suitable for large datasets

2. the possibility of using Apache Spark in order to evaluate the Davies-Bouldin Index and the Simplified Silhouette Index in a distributed manner.

The same metabolic pathways datasets from the previous work have been used in order to stress the application on structured data. The distributed Simplified Silhouette index showed very good sizeup performances (e.g., a 4-times larger dataset needs 3.5 to 4.6 times more time) and the speedup approaches the expected linear behaviour as the dataset size increases. The distributed Davies-Bouldin index outperforms the Simplified Silhouette index in terms of sizeup (e.g., a 4-times larger dataset needs 2.4 to 3.6 times more time) and showed an approaching linear behaviour for larger dataset as well. However, after 9 cores, the speedup curves started to flatten instead of increasing further: this is due to the fact that the Davies-Bouldin index has a lower computational complexity, thus fewer workers suffice. Indeed, the Davies-Bouldin index has linear complexity with respect to the number of patterns, viz $\mathcal{O}(N)$ which, when parallelised over p computational units, drops to $\mathcal{O}(N/p)$. Conversely, the Simplified Silhouette has complexity related to both patterns and medoids, that is $\mathcal{O}(kN)$ which, when parallelised, drops to $\mathcal{O}(kN/p)$.

6.3 Energy Management System Synthesis by ANFIS Networks

In smart grids, the Energy Management System is a module in charge of operating in real-time by defining the energy storage system energy flow in order to maximise the revenues generated by the energy trade with the distribution grid. Specifically, its correct functioning impacts the microgrid components (aggregated systems grouped in renewable sources power generators, electric loads and the energy storage system itself), whose flows must be managed and re-distributed across the microgrid and

from/to the main grid. Fuzzy systems have been proved successful in this regard, as thoroughly explored in literature [9, 68, 213–215, 239, 313] and in a first work [217] the energy management system has been synthesised thanks to an Adaptive NeuroFuzzy Inference System (ANFIS) [182] whose training leverages on clustering algorithms.

The ANFIS is equipped with multivariate Gaussian membership functions of the form

$$\phi^{(i)}(\mathbf{u}) = e^{-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}^{(i)})\cdot\mathbf{C}^{(i)^{-1}}\cdot(\mathbf{u}^T-\boldsymbol{\mu}^{(i)^T})} \tag{6.6}$$

where $\phi^{(i)}$ is the generic i^{th} membership function, defined by $\boldsymbol{\mu}^{(i)}$ and $\mathbf{C}^{(i)}$, namely the cluster centre and its covariance matrix, respectively, and $\mathbf{u}$ is the crisp input vector normalized in $[0, 1]$.

The ANFIS is defined by adopting the first order Takagi-Sugeno IF-THEN rules where the i^{th} membership function, $\phi^{(i)}$, defines the antecedent set of the i^{th} rule that reads as

$$\text{IF } \mathbf{u} \text{ is } \phi^{(i)} \text{ THEN } y^{(i)} = \mathbf{u}^T \cdot \gamma^{(i)} \tag{6.7}$$

where, in turn, $\gamma^{(i)}$ is the i^{th} rule consequent, defined by a properly tuned hyperplane coefficients vector. Due to the absence of AND-OR operators, the firing rule strength is exactly the membership function value, hence $f^{(i)} \equiv \phi^{(i)}$. A Winner-Takes-All strategy is adopted for computing the overall fuzzy inference system output as the output of the most firing rule (highest membership function value). From Eq. (6.6), it is clear that the membership function positions (and extend) can be tuned thanks to clustering algorithms, where $\boldsymbol{\mu}$ stands as the cluster representative (e.g., centroid) and $\mathbf{C}$ reads as the covariance matrix of the cluster itself. However recall that clustering algorithms notably work in the input space only, without relying on any output values: this makes standard clustering algorithms unsuitable for the problem at hand since it is impossible to evaluate the rule consequent (namely, the hyperplane). In order to overcome this problem, one must rely on the so-called *hyperplane clustering algorithms* [300] where the point-to-cluster distance has the form

$$\hat{d}(\langle \mathbf{x}, y\rangle, \langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle) = \epsilon d(\mathbf{x}, \boldsymbol{\mu}) + (1-\epsilon)(y - (\boldsymbol{\theta}^T\mathbf{x} - \theta_0))^2 \tag{6.8}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$ are the cluster centre and the hyperplane coefficients vector, respectively, $d(\cdot,\cdot)$ is a suitable point-to-centroid distance measure computed in the input space (e.g., squared Euclidean distance) and, finally, $\epsilon \in [0, 1]$ is a parameter weighting the two linear convex combination members. It is worth noting that the leftmost term, by considering patterns and centroids and by not considering output values or hyperplanes, can be seen as the distance in the *input space*. Conversely, the rightmost term considers output values and hyperplanes in lieu of centroids, therefore can be seen as the distance in the *hyperplanes space*. Joining the two terms, makes (6.8) a suitable *joint input-hyperplanes space* distance measure [217, 300], with ϵ in charge of weighting the input and hyperplanes spaces contributions. It is worth remarking that by setting $\epsilon = 0$ only the hyperplanes space is considered, thereby leading to a proper *hyperplane clustering*, and Eq. (6.8) becomes

$$\hat{d}(\langle \mathbf{x}, y\rangle, \boldsymbol{\theta}) = (y - (\boldsymbol{\theta}^T\mathbf{x} - \theta_0))^2 \tag{6.9}$$

By matching Eq. (6.8) and Eq. (6.9) it is clear that in hyperplane clustering centroids have no impact in the distance measure. Indeed, only the hyperplane coefficients vector $\boldsymbol{\theta}$ can be considered as the proper cluster representative. In turn,

hyperplanes are *affine subspaces* as they also include the fixed term, namely the intercept, as stressed by the term θ_0 in Eqs. (6.8) and (6.9). By following this approach, the cluster representative is just the coefficients vector $\boldsymbol{\theta}$, and the distance measure between a generic input-output pair $\langle \mathbf{x}, y \rangle$ with respect to a cluster is defined as the approximation error of the corresponding hyperplane on $\mathbf{x}$ (as in Eq. (6.9)).

In a first work, three candidate clustering algorithms have been considered (k-means, k-medians and k-medoids with the Mahalanobis distance) and tweaked in order to work on the input-hyperplane space (cf. Eq. (6.8)). Furthermore, four candidate values for $\epsilon = \{1, 0.75, 0.50, 0.25\}$ have been tested in order to show how the hyperplane space and the input space impact on the overall performances, measured by the profit generated by the energy trade with the main grid. The dataset provided by AReti SpA, the electricity distribution company in Rome (Italy), features profiles of buy and sell prices, energy exchanged with the grid, energy generation, energy demand and state of charge of the energy storage system. Tests showed that k-medoids is able to better minimise the profit ($\approx 15 - 20\%$ with respect to the benchmark solution formulated via mixed-integer linear programming).

The analysis has been extended in a second work [216], where the case $\epsilon = 0$ has been thoroughly discussed. There in indeed a potential issue with hyperplane clustering that needs to be carefully addressed: clusters well-separated in the input space are approximated by the same hyperplane. This leads to superpositions in the resulting membership functions, with undesired ambiguities. In order to overcome this problem, a (Pruning) Adaptive Resolution Min-Max Classifier [321, 322, 325] is used for membership function refining. As clustering algorithms are concerned, k-means and two hierarchical variants (agglomerative and divisive) are considered for comparison. Computational results on the same dataset showed that this procedure (hyperplane clustering supported by min-max classifier) yields better performances ($\approx 10 - 25\%$ with respect to the benchmark solution formulated via mixed-integer linear programming) if compared with the previous work.

A third work, finally, [218] compares different strategies for Energy Management System synthesis: Echo State Networks, Mamdani Fuzzy Inference System, ANFIS, MLP, SVR and Rolling Time Horizon equipped with two predictors ('ideal' predictor and Echo State Network). The Rolling Time Horizon strategy consists in running (at each time sample) the prediction algorithm to obtain estimations used by the optimisation algorithm in order to compute the optimal output (command) time series over a given time horizon, which is usually extended to one or two days. To better manage the prediction error, only the first element of the optimal time series is forwarded to the microgrid power converters in order to effectively balance the microgrid power flows. The price to pay for their good performances lies on its computational complexity. However, solutions based on soft computing (especially ANFIS-based) have competitive performances with respect to Rolling Time Horizon whilst featuring a much lower computational and structural complexity. Furthermore, ANFIS-based solutions can efficiently be implemented on low-cost hardware (e.g., a Raspberry Pi model B+) whilst keeping the decision time below 15 minutes[52], whereas the same is not true for Rolling Time Horizon-based strategies.

[52] In most European countries, generation group deviations are calculated on a 15 minutes basis and, consequently, energy systems are sampled every 15 minutes by assuming that the power generation and demand are constant.

Chapter 7

Conclusions

In this thesis, six different pattern recognition systems suitable for dealing with (hyper)graphs have been presented and investigated. These six systems somewhat span the entire three-fold partition of pattern recognition techniques from Chapter 3, namely feature engineering, embedding techniques and ad-hoc dissimilarities in the input space.

The first technique (Section 4.1), which falls under the feature generation umbrella, is a natural extension of [244], in which each graph is described according to its (sampled) spectral density. A second feature generation procedure relies on TDA (Section 4.2), by seeing graphs as purely topological objects and measuring the dissimilarity between their respective topological spaces thanks to their Betti numbers.

Then the dissertation moves towards embedding spaces, both explicit and implicit. The first (explicit) embedding technique still has roots in TDA, being based on simplicial complexes (Section 4.3). According to this technique, each simplex is a candidate information granule and, thanks to exact simplex matching, the total number of prospective granules can easily be shrunk. Further, this exact simplex matching (which is, in turn, possible thanks to the categorical nature of the node labels) makes the symbolic histogram evaluation very fast and straightforward. As common in GrC-based pattern recognition system, the model synthesis sees a suitable feature selection phase in order to further shrink the set of meaningful symbols for the problem at hand: this feature selection can either be performed in a 'naïve way' by means of genetic optimisation or in an automatic way (e.g., by ℓ_1 norm minimisation).

A second embedding procedure leverages on the INDVAL score (Section 4.4) in order to spot significant substructures in structures data (e.g., edges, paths in a graph). This is the only technique suitable for classification problems only, since the INDVAL evaluation (hence the embedding space definition) relies on the ground-truth class labels; the same is not true for all other techniques. The INDVAL helps in spotting relevant substructures which, like the simplices in the previous experiment, are candidate information granules for building the embedding space.

The family of four hypergraph kernels presented in Section 4.5 aim at defining suitable kernel functions in order to deal with hypergraphs (simplicial complexes) and they still heavily rely on the categorial nature of node labels.

Finally, in Section 4.6, a multiple kernel approach leveraging over multiple dissimilarity at the same time is presented. This technique not only can spot the most relevant dissimilarities (i.e., representations), but also suitable prototypes in the respective dissimilarity spaces, allowing a two-fold knowledge discovery phase.

Most of these techniques have been tested in order to solve a real-world problem, namely the enzymatic function prediction in protein networks (Section 5.2), with the two simplicial complexes-based techniques being the most performing ones. Further, the Betti numbers, the spectral density and the embedding over simplicial

complexes strategies have also been tested on another problem related to protein networks, that is the solubility degree prediction (Section 5.3). Also in this different problem (function approximation rather than classification), the embedding over simplicial complexes emerged as the most performing technique. Finally, this strategy has also been applied on a side problem, that is the classification of soluble vs. non-soluble proteins, again with remarkable results.

The INDVAL-based technique has been applied to the analysis of metabolic networks (Section 5.4) and a major comparison has been performed with a plain K-NN that considers the entire networks in order to quantify how much the semantic gap between the original space (graphs) and the embedding space (Euclidean) this technique is able to fill.

The three most cutting-edge techniques (the two embedding strategies and hypergraph kernels) have also been tested on benchmark datasets for graph classification and compared against suitable competitors (Section 5.5). Specifically, the four hypergraph kernels have been compared against four well-known competitors (WL, WLSP, PK and RW), with three of the four proposed kernels (WJK, SEK and EK) outperforming the competitors in the vast majority of the considered datasets. The embedding over simplicial complexes, as instead, has been compared against GRALG and against the WJK, with results close to the WJK. The INDVAL-based strategy has also been tested on benchmark data, yet it deserves a more in-depth comparison against existing techniques.

Needless to say, other than properly benchmarking the INDVAL-based embedding, the research here proposed paves the way for interesting research endeavours, especially as the extension of machine learning techniques towards hypergraphs is concerned. First, addressing how those techniques scale as the dataset size increases, possibly following the research notes in Appendix B. Second, and more interesting, extending the simplicial complexes-based techniques in order to consider also edge labels and more structured node labels, like GRALG does[53]. Hybridisations can also go the other way round, by equipping GRALG with an INDVAL-based prior filtering procedure before clustering the set of subgraphs in order to further reduce the computational complexity. Finally, the INDVAL strategy can easily be extended towards any type of structured data, for example text classification (spot relevant words, or n-grams within documents) or sequence analysis (spot relevant k-mers).

[53]GRALG performs an inexact graph matching between subgraphs and is capable of dealing with any type of node and/or edge labels. In order to make the comparison fair, GRALG did not consider edge labels either (dissimilarity between edges is constant) and has been equipped with a delta-distance between node labels (categorical).

Appendix A

Knowledge Discovery Data

A.1 Multiple Kernel Machines (EC number classification)

In the following tables, the sets of proteins elected as prototypes are listed. In order to shrink the output size, the a-posteriori analysis has been carried only on proteins which have been selected in all of the five runs of the genetic algorithm (in order to remove 'spurious' representatives due to randomness in the optimisation procedure).

TABLE A.1: Selected proteins in order to discriminate EC 1 (Oxidoreductases) vs. all the rest.

PDB ID	Notes / Description
1KOF	Transferase
1XFG	Transferase
3E2R	Oxydoreductase
4TS9	Transferase
1ZDM	Signalling Protein
1MPG	Hydrolase
1QQQ	Transferase

TABLE A.2: Selected proteins in order to discriminate EC 2 (Transferases) vs. all the rest.

PDB ID	Notes / Description
3EDC	LAC repressor (signalling protein)
1DKL	Hydrolase
1JKJ	Ligase
2DBI	Unknown function
3UCS	Chaperone
1LX7	Transferase
2GAR	Transferase
3ILI	Transferase
1S08	Transferase
4IXM	Hydrolase
4XTJ	Isomerase
1KW1	Lyase
1BDH	Transcription factor (DNA-binding)
4PC3	Elongation factor (RNA-binding)
5G1L	Isomerase

TABLE A.3: Selected proteins in order to discriminate EC 3 (Hydrolases) vs. all the rest.

PDB ID	**Notes / Description**
4RZS	Transcription factor (signalling protein)
1ZDM	Signalling protein
3I7R	Lyase
1HW5	Signalling protein
1SO5	Lyase

TABLE A.4: Selected proteins in order to discriminate EC 4 (Lyases) vs. all the rest.

PDB ID	**Notes / Description**
2BWX	Hydrolase
3UWM	Oxydoreductase
2H71	Electron transport
1D7A	Lyase
4DAP	DNA-binding
1SPV	Structural genomics, unknown function
1EXD	Ligase + RNA-binding
1X83	Isomerase
3ILJ	Transferase
2D4U	Signalling protein
1JNW	Oxydoreductase
1TRE	Oxydoreductase
1ZPT	Oxydoreductase
3LGU	Hydrolase
1IB6	Oxydoreductase
3C0U	Structural genomics, unknown function
5GT2	Oxydoreductase
2RN2	Hydrolase
4L4Z	Transcription regulator
3CMR	Hydrolase
1NQF	Transport protein
1GPQ	Hydrolase
4ODM	Isomerase + chaperone
2NPG	Transport protein
2UAG	Ligase
1OVG	Transferase
3AVU	Transferase
1RBV	Hydrolase
5AB1	Cell adhesion
1TMM	Transferase
4NIY	Hydrolase
4WR3	Isomerase

TABLE A.5: Selected proteins in order to discriminate EC 5 (Isomerases) vs. all the rest.

PDB ID	**Notes / Description**
4ITX	Lyase
2BWW	Hydrolase
5IU6	Transferase
1ODD	Gene regulatory
5G5G	Oxydoreductase
1G7X	Transferase
2E0Y	Transferase
2SCU	Ligase
1HO4	Hydrolase
3RGM	Transport Protein
1OAC	Oxydoreductase
5MUC	Oxydoreductase
3OGD	Hydrolase + DNA binding
4K34	Membrane protein
1Q0L	Oxydoreductase
1G58	Isomerase
5M3B	Transport protein
2WOH	Oxydoreductase
2PJP	Translation regulation (RNA-binding)

TABLE A.6: Selected proteins in order to discriminate EC 6 (Ligases) vs. all the rest.

PDB ID	**Notes / Description**
2OLQ	Lyase
1JDI	Isomerase
4NIG	Oxydoreductase + DNA-binding
5T03	Transferase
5FNN	Oxydoreductase
2Z9D	Oxydoreductase
2V3Z	Hydrolase
4ARI	Ligase + RNA-binding
3LBS	Transport protein
4QGS	Oxydoreductase
5B7F	Oxydoreductase
2ABH	Transferase

TABLE A.7: Selected proteins in order to discriminate not-enzymes vs. all the rest.

PDB ID	Notes / Description
1SPA	Transferase
2YH9	Membrane protein
1NQF	Transport protein
1LDI	Transport protein
1TIK	Hydrolase
1MWI	Hydrolase + DNA-binding
1GEW	Transferase
5CKH	Hydrolase
3ABQ	Lyase
3B6M	Oxydoreductase

A.2 Embedding Simplicial Complexes (EC number classification)

In Table A.9, the set of simplices that survived the feature selection phase for all five runs are listed. Each amino-acid composing the simplex is identified by its 1-letter code (see Table A.8). These simplices have been obtained with the following setup

- classifier: ℓ_1-SVMs
- sparsity weight in fitness function: $\alpha = 0.5$.

TABLE A.8: Amino-acids nomenclature table.

Amino-acid	3-letter code	1-letter code
alanine	ala	A
arginine	arg	R
asparagine	asn	N
aspartic acid	asp	D
asparagine or aspartic acid	asx	B
cysteine	cys	C
glutamic acid	glu	E
glutamine	gln	Q
glutamine or glutamic acid	glx	Z
glycine	gly	G
histidine	his	H
isoleucine	ile	I
leucine	leu	L
lysine	lys	K
methionine	met	M
phenylalanine	phe	F
proline	pro	P
serine	ser	S
threonine	thr	T
tryptophan	trp	W
tyrosine	tyr	Y
valine	val	V

TABLE A.9: Selected simplices for EC number classification.

EC1		**EC2**		**EC3**		**EC4**	**EC5**	**EC6**	**not-enzymes**	
G-S-W	L-T-V	C-L-R	K-K-L	K-L-W	G-P-S	E-S-V	A-A-V	F-L-V	K-T-T	L-T
A-I-S	L-L-V	H-R-T	G-L-Q	A-L-V-V	A-M	G-V-V	A-G-N	D-G-T	E-G-I	A-L-S
G-R-S	A-R	I-T-V	E-R-T	F-S	D-N-S	A-C-V	D-L-L	D-N-S	F-G-V	G-I-L
G-L	A-C-L	C-H-V	A-L-Q	A-E-T	D-F-L	A-H-L	D-L-T	E-F-V	N-V-Y	E-K-P
A-H-I	L-Q-Q	A-I-K	A-F-R	D-K-R	A-K	A-N	G-G-S	K-L	D-K-R	D-T-T
E-F-S	V-V-Y	A-Y	Q-T	G-L-T	G-L-V	E-F-K	A-V-V	F-G-I	N-V-V	A-A-E
Q-S	E-K-T	A-I-M	K-P-V	A-A-G-V	E-G-I	E-K	E-L-Q	F-I-P	V-V-V	G-L-S
H-Q-R	D-L-V	A-A	A-F-I	G-I-K	K-Q-V	A-I-I	I-R-T	D-G-G	H-Q	G-N-Y
F-P	I-V-V	D-M	G-L-R	D-D	G-T-Y	A-Q-R	F-G-V	D-G-L	D-L-S	G-G-N
D-I-T	A-F-G	A-A-A-V	E-G-S	G-I-L	G-G-N	A-D-N	V-V-V	E-L-P	G-I-T	E-K-N
F-L	G-V-Y	A-M-P	L-T	S-V	C-G-L	A-I-L	H-H	E-R-T	I-P-R	K-R
P-S	A-I-M	K-P	G-H-L	D-K-L	A-D	L-R-V	R-T-V	I-I-S	D-G-I	A-G-N
D-G-I	G-P-V	D-R-V	D-G-I	D-H-T	C-L-V	T-V-V	F-R-S	G-K-R	L-N-V	G-G-M
A-G-L	E-N-V	A-I-Y	A-G-T	E-S-V	F-G-N	I-P-S	A-R-V	L-M-V	E-G-K	K-L-Y
F-G-M	A-F-T	L-P-P	A-G-V	A-D-L	E-R-W	L-R	L-P-V	A-D-I	F-L-L	A-Q-V
A-A-R	P-R	N-P-R	A-G-S	F-I-I	F-F-G	A-F-Y	E-I-V	G-I-V	R-V-Y	A-A-I-V
A-A-L-V	A-F-W	L-P-S	K-Q-V	K-R-W	A-D-I	A-P-V-V	A-L-V	G-I-S	D-K-S	L-L-Q
T-V-Y	A-D-E	D-I	L-N-R	G-N-Y	A-E-H	E-I-P	G-G-V	D-I	D-K-L	N-R-S
I-I-N	D-E	A-E-I	E-E	D-E-P	D-F-N	I-L-W	E-F-K	L-P-Q	G-P-R	E-Q-S
A-A-K	C-D-T	L-N	D-D	D-R-W	F-F-L	A-G-L	A-G-L	L-R	A-A-G-G	A-L
A-E-G	A-N-Q	A-I-R	I-T-T	E-Q		D-L-R	F-L-L	I-T-Y	D-I-V	E-V-V
L-Q-W	D-N	F-P-Q	F-K-V	N-Q-V		F-L-V	D-L	E-V-Y	Q-T	N-S-V
A-A-G	E-F-G	G-L-Y	I-K-L	E-L		A-S-S	I-V-W	L-R-S	Q-R	A-E
N-Q-V	A-N-V	A-L-N	H-R	E-G		L-Q-S	L-Q-V	A-A-G-V	K-K-L	A-G
D-H-Y	I-N-P	S-V	T-V-Y	S-V-Y		K-R	A-D-V	G-S	A-T-Y	I-T-V
G-V-V		A-G	K-L	A-H-M		D-D		A-S-Y	D-Q-V	K-L-R
D-H-R		I-I-P	A-K-P	E-P-R		D-G-I		G-K-V	F-G-T	E-V-Y
A-E-T		G-L-T	I-S-Y	E-S-T		K-L-R		D-N-T	K-S-V	A-G-W
P-V-V		S-S-T	A-E-Y	A-M-V		D-L		D-I-R	H-N-V	A-C-V
F-I-Y		I-L-V	G-I-K	S-T		L-S-V		L-L-P	A-H-M	I-I-V
H-P-R		A-N	I-L-P	F-I-S		G-T-T		G-I-T	A-F-G	F-M-V
D-F-T		V-V-Y	A-E-T	A-F-R		P-V-Y		A-K-S	F-L-R	S-V-Y
L-R-V		A-C-D	P-S	D-D-V		I-N-V		P-V	A-H-P	G-R
L-P-V		F-M-V	E-F-Q	G-T-V		A-E-Y		F-G-G	I-Q-V	I-L-N
K-R		I-L-L	I-L-S	A-F-P		E-N-R		K-R	L-N-P	E-L-S
I-P-V		A-C-I	A-H-L	A-A-Q		I-V-Y		G-G-I	E-K-Q	E-L-R
E-L-T		D-F-K	G-I-V	A-Q-Y		A-E-T		I-I-V	R-S-S	A-G-Y
E-I-V		F-P-Y	L-Q-V	A-Q-R		A-A-S		K-L-R	L-L-V	I-L-M
E-L-V		Q-R-T	F-I-S	I-I-L		G-L-S		E-L-R	G-I-R	G-G-T
E-L-S		G-T-T	K-T	A-H-Y		A-E-K		E-I-S	A-I-R	A-F-R
A-L-T		A-Q-W	H-K-W	A-D-H		S-T-V		L-L-R	F-I-Y	E-F-G
I-L-R		P-V-V	A-K-S	E-Q-R		H-L-Q		G-T-Y	G-K-N	C-H-V
G-H-T		E-V-V	A-D-E	L-L-V		I-V-V		E-I-K	G-I-Q	I-V-V
E-G-I		K-L-R	H-L-L	I-M-T		A-K-M		A-L-V	H-K	G-T-V
G-L-Q		A-I-W	L-S	T-V-V		I-S-V		F-I-L	A-S-V	V-V-Y
F-L-L		A-L-L	C-L-L	Q-R-T		A-E-G		A-I-V	D-F-V	D-R-V
H-L-P		D-E	E-L-P	S-T-W		H-L-S		D-L-T	H-V-Y	D-I
A-C-Q		F-Q-S	H-L-N	P-S-V		I-I-V		A-E-G	E-R	G-I-S
I-T-V		E-N	C-L-P	D-Q-V		H-L-T		D-K-Q	A-A-A	S-T
L-P-S		F-G-L	T-T-T	T-V-W		A-A-I		G-T	G-V-V	D-V-Y
I-I-L		A-A-V	A-E-V	F-I-T		L-L		R-T-V	K-L-Q	D-R
E-N		I-T	A-K-Y	F-L-Y		L-L-R		D-Q-S	A-A-V	G-L-T
E-K		E-W	L-M-R	A-C-I		A-A-V		S-V-W	A-A-R	P-T
E-T		E-L-S	G-I-R	I-Q-Y		D-L-Q		G-K	D-G-L	N-Y
A-S-V		D-I-I	L-M-T	A-T-V		P-S-V		L-Q-W	L-Q	L-T-W
A-L-Q		F-I-T	D-Q	H-K-L		D-Q-S		A-E-K	P-Q-T	A-D-T
R-T-V		I-I	I-R-R	S-S		F-G-L			A-F-W	A-D-E
A-A		A-A-I-V	D-D-L	D-I-V		G-H-P			A-L-L	C-L-Y
I-I-S		A-L-L-V		I-L		I-L-R			I-L-V	A-D-I
L-L-L		F-N-N		I-K-L		L-L-Q			G-N	L-R-S
L-L-T		D-G-N		F-I-N		A-S-T			Q-S	A-R
L-R		G-R		A-F-K		L-R-T			G-K	A-G-M
I-R-R		A-T-Y		G-K-V		A-L-N			I-L-Q	A-D-F
L-M-V		D-G-V		A-L-S		D-E-I			D-L-R	L-Q-V
D-I-Y		D-G-R		G-G-V					A-F-S	I-K-T
I-K-T		E-P		L-P-V					A-L-P	E-N-V
I-N-V		A-R-V		E-I-S					A-L-Q	D-E
D-R-V		E-L		L-P-S					A-L-V	A-N-R
E-K-Q		A-A-T		A-G-T					F-F-L	P-V-V
A-Q-V		G-N-Q		A-H-P					D-L-W	D-P-Q
G-P-T		A-A-R		D-L-N					A-G-I	

A.3 Embedding Simplicial Complexes (solubility classification)

Recalling Section 5.3.4, the proposed embedding technique over simplicial complexes with ℓ_1-SVMs found the best solubility threshold in range $\tau \in [0.5, 0.7]$. In the following Table, the set simplices that survived the feature selection phase for all of the five Training-Validation-Test splits for all of the candidate values $\tau = 0.5$, $\tau = 0.6$ and $\tau = 0.7$ are shown.

TABLE A.10: Selected simplices for solubility classification.

Simplex
E-L
G-Q-V
A-P-W
D-E
E-S-V
E-E-R
E-G-L
E-R-S
F-M-V
L-T-V
A-L-W
A-F-L
A-A-H
G-L-P
G-I-T
D-D-L
A-E-V
A-L-T
A-I-R
A-R-V
A-G-Q
L-S
I-L-P
D-N-Q
F-F-P
I-L-S
P-R-V
I-Q
A-A-A

Appendix B

Notes on Parallel and Distributed Evaluations

The amount of data and information available nowadays made *Big Data* a buzzword in the information science and computer science communities alike. So, one might ask whether the proposed techniques are able to scale for large datasets. For the sake of shorthand, the discussion relies only on building the feature vectors (either by using a feature engineering approach or by using an embedding technique) and does not concern the classification and optimisation task. The rationale behind this hypothesis is that the entire (structured) dataset might not fit in memory or the generation of such feature vectors is too expensive (computationally speaking) to be done in a serial manner whilst keeping reasonable running times. These parallel implementation have been proved to be effective (i.e., they return the same solution as the non-parallel counterpart), yet their efficiency (e.g., in terms of sizeup and speedup) must be further investigated.

B.1 Feature Generation using Graphs Spectral Density

The spectral density-based technique described in Section 4.1 is embarrassingly parallel due to the independence of the feature vectors generation. Indeed, recall that (for each graph) one shall evaluate the adjacency matrix, the degree matrix, the normalised Laplacian matrix, the spectrum of the Laplacian matrix, its spectral density and, finally, draw a finite number of samples from such spectral density. There are two ways in which one can leverage on distributed systems and the choice strictly depends on the size of the available networks. The computational bottleneck of this feature generation chain is the evaluation of the spectrum (eigenvalues) of the normalised Laplacian matrix.

If the networks are quite big (in terms of number of nodes) GPUs might be of help in speeding up the feature vectors generation. Given the adjacency matrix $\mathbf{A}$, both the evaluation of $\mathbf{D}$ and the evaluation of $\overline{\mathbf{L}}$ fully rely on matrix operations whose GPU acceleration is straightforward. Furthermore, there exist efficient linear algebra libraries for GPUs which can also take care of evaluating the eigenvalues of $\overline{\mathbf{L}}$. The kernel density estimator and final sampling can be performed on the CPU, since their computational effort is negligible with respect to the previous steps. The features generation phase can be performed in a serial manner, with a plain for-loop iterating over the available networks and exploiting the GPU for processing network. Recalling that $\mathbf{A}$, $\mathbf{D}$ and $\overline{\mathbf{L}}$ scale quadratically in terms of number of nodes, the problem here is not 'having too many networks'. Rather, the problem is that processing a single network takes a huge amount of time, so the speedup one can obtain has to be imputed to the acceleration given by the GPU on each network.

Let us now consider the opposite scenario, in which there is a consistent number of networks and their sizes are rather small. If the networks are small, the time required for solving the eigenvalues problem on the CPU and on the GPU tend to be similar, so the previous serial for-loop will unlikely be helpful. So, one can exploit multiple cores available on the CPU and perform a parallel for-loop over the available networks where each thread (core) takes care of processing a given network.

Obviously, depending on the framework at hand, hybrid approaches can also be pursued where multiple CPU threads in parallel exploit the GPU. The main limit of this hybrid approach is the on-board GPU memory: the number of concurrent threads and the data they need in order to accomplish their tasks must fit in the GPU memory, otherwise an inevitable out-of-memory error aborts the execution.

B.2 Feature Generation using the Betti Numbers

As typical in feature generation approaches, there is a chain of processing stages that individually involves each pattern. In this case, the processing chain concerns the evaluation of a simplicial complex, the evaluation of boundary operators at different simplicial orders and their Betti numbers. The computational bottleneck here is another linear algebra operation, namely the matrix rank, hence similar observations hold with respect to the previous case, where the computationally expensive algebraic operator was the eigenvalue solver.

B.3 Embedding over Simplicial Complexes

Things start getting more interesting as embedding techniques are concerned. Indeed, conversely to feature generation scenarios where a chain of processing stages are individually applied to each of the available patterns, building an embedding space usually needs some sort of 'cooperation' between patterns.

Here, a MapReduce-based workflow for building the embedding space over simplicial complexes is sketched.

Let us start by presenting a MapReduce workflow in order to synthesise the alphabet $\mathcal{A}$ where, for the sake of generalisation, each pattern is a node-labelled graph. In the Map phase (summarised in Algorithm 1), each worker processes a given pattern drawn from $\mathcal{D}_{TR} \cup \mathcal{D}_{VAL}$, performing the following tasks:

1. infers the simplicial complex starting from the 1-skeleton (e.g., thanks to the Clique complex)
2. by exploiting node labels, identifies each vertex belonging to each simplex according to its node label
3. removes duplicates
4. returns $\langle$key, value$\rangle$ pairs where keys are simplices and values are dummy 1's.

After the MapReduce framework automatically shuffles output $\langle$key-value$\rangle$ pairs and groups them by keys, the Reduce phase acts as follows (Algorithm 2):

1. sum the 1's belonging to the same keys
2. return the key (simplex) and the value (sum of occurrences)

Algorithm 1: Pseudocode for the alphabet synthesis: Map phase.

Input : Data shard composed by a given number of graphs drawn from $\mathcal{D}_{TR} \cup \mathcal{D}_{VAL}$
Output: Intermediate $\langle$key, value$\rangle$ pairs

```
/* Start evaluating the simplicial complexes                    */
SC = list();                                  // list of simplicial complexes
for each graph G in shard do
    SC.append(CliqueComplex(G));
end
/* Identify simplices by node labels                            */
for each simplicial complex S ∈ SC do
    for each node u ∈ S do
        u := L_v(u);
    end
end
/* Remove duplicates                                            */
S = flatten(SC);          // list of simplices, regardless of their complex
for each simplex s ∈ S do
    s := sort(s);                              // sort nodes within simplex
end
S = unique(S);
/* Flush key-value pairs                                        */
for each simplex s ∈ S do
    Emit(key=s, value=1);
end
```

Algorithm 2: Pseudocode for the alphabet synthesis: Reduce phase.

Input : Key k, list of its values $[v_1, \ldots, v_n]$
Output: Key k, sum of its values s

```
s = 0;
for each value v do
    s = s + v;
end
/* Flush final results                                          */
Emit(key=k, value=s);
```

The sum of occurrences can deliberately be discarded, being it just a trick to return the unique keys (simplices), which will form the alphabet.

The proper embedding can also be performed in a MapReduce fashion. Along with the Map phase (Algorithm 3), the alphabet $\mathcal{A}$ must be broadcasted to all workers. Then, in the Map phase, each worker processes a given pattern drawn from $\mathcal{D}$, performing the following tasks:

1. infers the simplicial complex starting from the 1-skeleton (e.g., thanks to the Clique complex)

2. by exploiting node labels, identifies each vertex belonging to each simplex according to its node label

3. allocates an array with length $|\mathcal{A}|$
4. scan the simplicial complex from steps 1–2 and fill the array
5. return key-value pair with the ID of the graph as key and the array as value

The Reduce phase (Algorithm 4) is the so-called 'identity reducer', as it receives key-value pairs and writes them on disk.

Algorithm 3: Pseudocode for the embedding procedure: Map phase.

Input : Data shard composed by a given number of graphs drawn from $\mathcal{D}$, alphabet $\mathcal{A}$
Output: Intermediate $\langle$key, value$\rangle$ pairs

```
   /* Start evaluating the simplicial complexes                    */
 1 SC = list();                          // list of simplicial complexes
 2 for each graph 𝒢 in shard do
 3 |   SC.append(CliqueComplex(𝒢));
 4 end
   /* Identify simplices by node labels                            */
 5 for each simplicial complex 𝒮 ∈ SC do
 6 |   for each node u ∈ 𝒮 do
 7 |   |   u := ℒ_v(u);
 8 |   end
 9 end
   /* Symbolic histograms                                          */
10 for i = 1, ..., |SC| do
11 |   h = zeros(1 × |𝒜|);
12 |   for j = 1, ..., |𝒜| do
13 |   |   h_j = count(𝒜_j, SC_i);
14 |   end
15 |   Emit(key=i, value=h);                  // Flush key-value pairs
16 end
```

Algorithm 4: Pseudocode for the embedding procedure: Reduce phase.

Input : Key k, its value $\mathbf{h}$
Output: Key k, its value $\mathbf{h}$

```
   /* Flush final results                                          */
 1 Emit(key=k, value=h);
```

Obviously one might object that the simplicial complex must be evaluated in both Map phases, resulting in a waste of computational efforts. In order to overcome this problem, one can start with a dataset already composed by simplicial complexes rather than composed by 1-skeletons. In order to do this, one must add an additional MapReduce job before the alphabet synthesis where the Map phase infers the simplicial complex and the Reduce phase writes it on disk. In this way, Lines 1–4 (or Lines 1–9 if one wants also to consider the node labelling) from both the Map phases (Algorithms 1 and 3) can be avoided.

B.4 Embedding via INDVAL

Recall from Section 6.2 the major differences between MapReduce and Apache Spark. The atomic data structure in Spark is the so-called Resilient Distributed Dataset (RDD) [399]: distributed (across workers[54]) collection of data with fault-tolerance mechanisms, which can be created starting from many sources (distributed file systems, databases, text files and the like) or by applying *transformations* on other RDDs. Example of transformations[55] which will turn useful in the following are:

map(): `rdd2 = rdd1.map(f)`
creates `rdd2` by applying function f to every element in `rdd1`

join(): `rdd3 = rdd2.join(rdd1)`
creates `rdd3` containing pairs with matching keys from `rdd1` and `rdd2`

filter(): `rdd2 = rdd1.filter(pred)`
creates `rdd2` by filtering elements from `rdd1` which satisfy predicate *pred* (i.e., if *pred* is *True*)

union(): `rdd3 = rdd1.union(rdd2)`
creates `rdd3` by vertically stacking `rdd1` and `rdd2`

flatMapValues(): `rdd2 = rdd1.flatMapValues(f)`
creates `rdd3` by first applying a function f to all values of `rdd1`, and then flattening the results (note: the RDD must have ⟨key, value⟩ pair-like records)

cartesian(): `rdd3 = rdd1.cartesian(rdd2)`
creates `rdd3` by evaluating the cartesian product between items in `rdd1` and `rdd2`

reduceByKey(): `rdd2 = rdd1.reduceByKey(f)`
creates `rdd2` by merging, according to function f, all values for each key in `rdd1`: `rdd2` will have the same keys as `rdd1` and a new value obtained by the merging function.

Similarly, examples of *actions* which can be applied to RDDs are:

count(): `rdd.count()`
count the number of elements in `rdd`

collect(): `rdd.collect()`
collects in-memory on the master node the entire `rdd` content

toLocalIterator(): `rdd.toLocalIterator()`
returns an iterator that scans all items in `rdd` in a partition-wise manner

collectAsMap(): `rdd.collectAsMap()`
returns in-memory on the master node a dictionary storing the entire `rdd` content (note: the RDD must have ⟨key, value⟩ pair-like records).

[54] In Spark, *workers* are the elementary computational units, which can be either single cores on a CPU or entire computers, depending on the environment configuration (i.e., the *Spark Context*).

[55] For a more extensive list, the interested reader is referred to the official Apache Spark RDD API at https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

Let us suppose to have three RDDs containing the Training Set (`rdd_tr`), the Validation Set (`rdd_val`) and the Test Set (`rdd_ts`), respectively. Each record in these RDDs contains a unique identifier, the graph $\mathcal{G}$ and its problem-related class (recall that the INDVAL strategy needs the ground truth class values for scoring edges' sensitivity and specificity), hence:

$$\langle \text{ ID}, \mathcal{G}, l \rangle$$

The first step is to evaluate, for each graph in each RDD, its edge list (or path list, in a more general sense). This can be accomplished thanks to a `map()` function that strips $\mathcal{G}$ from each record and returns the list of its paths $\mathcal{W}$, eventually expressed by means of the node labels at their extremities. At the end of the `map()` function, $\mathcal{G}$ can be dropped from the RDD records for the sake of memory footprint: indeed, its paths are of interest as the INDVAL score is concerned. After this `map()` phase, each record from the three RDDs will have the form:

$$\langle \text{ ID}, \mathcal{W}, l \rangle$$

Now the INDVAL must be evaluated, so a temporary RDD (`rdd_tv`) containing Training and Validation Set can be build thanks to `union()`:

$$\texttt{rdd\_tv} = \texttt{rdd\_tr.union(rdd\_vl)}$$

This RDD should not be cached in memory on the available workers (it would be a waste of space as it can be easily generated if the three original RDDs are cached). A chain of transformations can be pipelined to this RDD in order to return all the desired actors.

Let us start by evaluating the denominator for B (see Eq. (4.17)), namely the number of patterns per class: this can be done by a simple MapReduce-like job over `rdd_tv`:

$$\begin{gathered}\texttt{den\_B} = \texttt{rdd\_tv.map(}\langle \text{ ID}, \mathcal{W}, l \rangle \rightarrow \langle key = l, value = 1 \rangle \texttt{)} \setminus \\ \texttt{.reduceByKey(}\ f = sum(v_1, v_2)\ \texttt{).collectAsMap()}\end{gathered}$$

The `map()` phase strips the label l from each record (which serves as the key) and a 1 is considered as its corresponding value. The `reduceByKey()()` phase sums the 1's corresponding to each key (i.e., label) and, finally, `collectAsMap()` returns on the master node a dictionary containing the number of items per label. Since this dictionary has as many entries as there are classes, it can be safely collected on the master node with no risk of memory blow-up.

Let us continue by evaluating the numerator for both A and B (see Eqs. (4.16)–(4.17)), namely the number of times a given path appears in patterns belonging to different classes:

$$\begin{gathered}\texttt{num\_rdd} = \texttt{rdd\_tv.map(}\langle \text{ ID}, \mathcal{W}, l \rangle \rightarrow \langle key = l, value = \mathcal{W} \rangle \texttt{)} \setminus \\ \texttt{.flatMapValues(}\ f(x) = x\ \texttt{).map(}\ \langle l, w \in \mathcal{W} \rangle \rightarrow \langle key = \{l, w\}, value = 1 \rangle \texttt{)} \setminus \\ \texttt{.reduceByKey(}\ f = sum(v_1, v_2)\ \texttt{)}\end{gathered}$$

Let us start from `rdd_tv` and, using a `map()` function, let us reshape the RDD in such a way that l serves as the key and $\mathcal{W}$ serves as the value. However, $\mathcal{W}$ is a list of paths, so the RDD should be flattened in such a way that each label is associated to a path rather than a list of paths: this can be done thanks to `flatMapValues()` with a dummy flattening function. The path along with the label serve as the key for another MapReduce-like job in which a 1 are considered as its corresponding value. Another `reduceByKey()` task takes care of summing the 1's so, at the end of

this stage, we have the counts (values) of how many times a path is associated to a class (keys). The resulting RDD (`num_rdd`) cannot be collected in memory, as it grows with the number of paths. Furthermore, one might think that the flattening procedure might result in a memory blow-up even in distributed scenarios since this procedure augments in a non-negligible way the number of records in the RDD: however, by pipelining further transformations, in no point of the execution the flattened RDD is fully present in memory.

Finally, one shall evaluate the denominator for A, namely the number of times a path appears, regardless of the class. Recall that each record from `num_rdd` has the form

$$\langle \{w, l\}, \text{count} \rangle$$

then, simply

$$\texttt{den\_A} = \texttt{num\_rdd.map(}\ \langle \{w, l\}, \text{count} \rangle \rightarrow \langle key = w, value = \text{count}\ \rangle\ \texttt{)}\ \backslash$$
$$\texttt{.reduceByKey(}\ f = sum(v_1, v_2)\ \texttt{)}$$

In order to count the number of times a path appears, one can get consider only the path w to be the key for the MapReduce-like job and discard the stratification given by the class l. Thanks to the `map()` function one can reshape the record as $\langle w, \text{count} \rangle$ and trigger another `reduceByKey()` task which sums the counts related to the same w (path). This RDD (`den_A`) cannot be collected in memory either, since its size is related to the number of paths.

Now that all actors are in place, the RDDs containing A, B and I for each path and for each class can be evaluated as follows:

1. A (`A_rdd`) can be obtained by joining `num_rdd` and `den_A` using the $\{w, l\}$ pair as the key
2. B (`B_rdd`) can be obtained by broadcasting `den_B` to all workers and then, using a $map()$ function, the value associated to each record from `num_rdd` can be divided by the corresponding value from `den_B` by paying attention at the class l
3. I (`I_rdd`) can be obtained by joining `A_rdd` and `B_rdd` by using the $\{w, l\}$ pair as the key.

Finally, by means of a filtering operation, one can collect all symbols whose INDVAL score is greater than (or equal to) a given threshold T, that is:

$$\texttt{Alphabet} = \texttt{I\_rdd.filter(}\ \langle \{w, l\}, I \rangle : I \geq T\ \texttt{).collect()}$$

The variable `Alphabet` is a list-like variable that contains all symbols composing the alphabet (i.e., whose INDVAL score is above the desired threshold T). Obviously, the major assumption is that this list can be safely collected on the master node memory.

The embedding procedure follows from Algorithm 3: the alphabet is broadcasted to all workers and each worker, in parallel, evaluates the symbolic histograms of a given dataset shard with respect to the alphabet.

Bibliography

[1] B. Alberts et al. *Essential cell biology*. Garland Science, 2015.

[2] B. Alberts et al. *Molecular Biology of the Cell*. 4th ed. New York, USA: Garland Science, 2002. ISBN: 0815332181.

[3] E. Aldea, J. Atif, and I. Bloch. "Image Classification Using Marginalized Kernels for Graphs". In: *Graph-Based Representations in Pattern Recognition*. Ed. by F. Escolano and M. Vento. Springer, Berlin, Heidelberg, 2007, pp. 103–113. ISBN: 978-3-540-72903-7. DOI: 10.1007/978-3-540-72903-7_10.

[4] L. A. N. Amaral et al. "Classes of small-world networks". In: *Proceedings of the National Academy of Sciences* 97.21 (2000), pp. 11149–11152. ISSN: 0027-8424. DOI: 10.1073/pnas.200327197.

[5] K. Anand and G. Bianconi. "Entropy measures for networks: Toward an information theory of complex topologies". In: *Phys. Rev. E* 80 (4 2009), p. 045102. DOI: 10.1103/PhysRevE.80.045102.

[6] W. N. Anderson Jr. and T. D. Morley. "Eigenvalues of the Laplacian of a graph". In: *Linear and Multilinear Algebra* 18.2 (1985), pp. 141–145. DOI: 10.1080/03081088508817681.

[7] C. B. Anfinsen. "Principles that Govern the Folding of Protein Chains". In: *Science* 181.4096 (1973), pp. 223–230. ISSN: 0036-8075. DOI: 10.1126/science.181.4096.223.

[8] M. Ankerst et al. "OPTICS: ordering points to identify the clustering structure". In: *ACM Sigmod record*. Vol. 28. 2. ACM. 1999, pp. 49–60.

[9] D. Arcos-Aviles et al. "Low complexity energy management strategy for grid profile smoothing of a residential grid-connected microgrid using generation and demand forecasting". In: *Applied Energy* 205 (2017), pp. 69–84. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.07.123.

[10] M. A. Armstrong. *Basic Topology*. Springer-Verlag New York, 1983. DOI: 10.1007/978-1-4757-1793-8.

[11] D. Arthur and S. Vassilvitskii. "K-means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics. Philadelphia, USA, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5.

[12] M. Artin. *Algebra*. Prentice Hall, 1991. ISBN: 9780130047632.

[13] F. R. Bach. "Graph Kernels Between Point Clouds". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM. New York, NY, USA, 2008, pp. 25–32. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390160.

[14] F. R. Bach, G. R. Lanckriet, and M. I. Jordan. "Multiple kernel learning, conic duality, and the SMO algorithm". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 6. DOI: 10.1145/1015330.1015424.

[15] G. Bagler and S. Sinha. "Network properties of protein structures". In: *Physica A: Statistical Mechanics and its Applications* 346.1 (2005). Statphys - Kolkata V: Proceedings of the International Conference on Statistical Physics: 'Complex Networks: Structure, Function and Processes', pp. 27 –33. ISSN: 0378-4371. DOI: 10.1016/j.physa.2004.08.046.

[16] L. Baldini, A. Martino, and A. Rizzi. "Exploiting Cliques for Granular Computing-basedGraph Classification". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Under Review. 2020.

[17] L. Baldini, A. Martino, and A. Rizzi. "Stochastic Information Granules Extraction for Graph Embedding and Classification". In: *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: NCTA, (IJCCI 2019)*. INSTICC. SciTePress, 2019, pp. 391–402. ISBN: 978-989-758-384-1. DOI: 10.5220/0008149403910402.

[18] L. Baldini, A. Martino, and A. Rizzi. "Towards a Class-Aware Information Granulation for Graph Embedding and Classification". In: *Computational Intelligence: 11th International Joint Conference, IJCCI 2019 Vienna, Austria, September 17-19, 2019 Revised Selected Papers*. To appear in.

[19] L. Baldini et al. "Complexity vs. Performances in Granular Embedding Spaces for Graph Classification". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Under Review. 2020.

[20] J. R. Banavar and A. Maritan. "Physics of Proteins". In: *Annual Review of Biophysics and Biomolecular Structure* 36.1 (2007), pp. 261–280. DOI: 10.1146/annurev.biophys.36.040306.132808.

[21] J. R. Banavar and S. Vishveshwara. "Protein structure and folding – simplicity within complexity". In: *Journal of Biomolecular Structure and Dynamics* 31.9 (2013), pp. 973–975. DOI: 10.1080/07391102.2012.748533.

[22] H.-J. Bandelt and V. Chepoi. "Metric graph theory and geometry: a survey". In: *Contemporary Mathematics* 453 (2008), pp. 49–86.

[23] H.-J. Bandelt and E. Prisner. "Clique graphs and Helly graphs". In: *Journal of Combinatorial Theory, Series B* 51.1 (1991), pp. 34 –45. ISSN: 0095-8956. DOI: 10.1016/0095-8956(91)90004-4.

[24] Y. M. Bar-On, R. Phillips, and R. Milo. "The biomass distribution on Earth". In: *Proceedings of the National Academy of Sciences* 115.25 (2018), pp. 6506–6511. ISSN: 0027-8424. DOI: 10.1073/pnas.1711842115.

[25] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pp. 509–512. ISSN: 0036-8075. DOI: 10.1126/science.286.5439.509.

[26] A.-L. Barabási and E. Bonabeau. "Scale-free networks". In: *Scientific american* 288.5 (2003), pp. 60–69.

[27] S. Barbarossa, S. Sardellitti, and E. Ceci. "Learning from Signals Defined over Simplicial Complexes". In: *2018 IEEE Data Science Workshop (DSW)*. 2018, pp. 51–55. DOI: 10.1109/DSW.2018.8439885.

[28] S. Barbarossa and M. Tsitsvero. "An introduction to hypergraph signal processing". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 6425–6429. DOI: 10.1109/ICASSP.2016.7472914.

[29] A. Bargiela and W. Pedrycz. *Granular computing: an introduction*. Kluwer Academic Publishers, Boston, 2003.

[30] M. H. Barley, N. J. Turner, and R. Goodacre. "Improved descriptors for the quantitative structure–activity relationship modeling of peptides and proteins". In: *Journal of chemical information and modeling* 58.2 (2018), pp. 234–243. DOI: 10.1021/acs.jcim.7b00488.

[31] A. Barrat et al. "The architecture of complex weighted networks". In: *Proceedings of the National Academy of Sciences* 101.11 (2004), pp. 3747–3752. ISSN: 0027-8424. DOI: 10.1073/pnas.0400087101.

[32] M. Barthélemy. "Spatial networks". In: *Physics Reports* 499.1 (2011), pp. 1 – 101. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2010.11.002.

[33] A. E. Bartz. *Basic Statistical Concepts*. New York: Macmillan Pub Co., 1988.

[34] A. Bashan et al. "Network physiology reveals relations between network topology and physiological function". In: *Nature communications* 3 (2012), p. 702. DOI: 10.1038/ncomms1705.

[35] V. Beckers et al. "In silico metabolic network analysis of Arabidopsis leaves". In: *BMC systems biology* 10.1 (2016), p. 102. DOI: 10.1186/s12918-016-0347-3.

[36] J. M. Berg, J. L. Tymoczko, and S. Lubert. *Biochemistry*. 5th ed. New York, USA: W. H. Freeman, 2002. ISBN: 0-7167-3051-0.

[37] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[38] H. M. Berman et al. "The Protein Data Bank". In: *Nucleic Acids Research* 28.1 (2000), pp. 235–242.

[39] M. J. Berridge. "The inositol trisphosphate/calcium signaling pathway in health and disease". In: *Physiological reviews* 96.4 (2016), pp. 1261–1296. DOI: 10.1152/physrev.00006.2016.

[40] F. M. Bianchi, L. Livi, and A. Rizzi. "Two density-based k-means initialization algorithms for non-metric data clustering". In: *Pattern Analysis and Applications* 19.3 (2016), pp. 745–763. ISSN: 1433-755X. DOI: 10.1007/s10044-014-0440-4.

[41] F. M. Bianchi et al. "A Granular Computing approach to the design of optimized graph classification systems". In: *Soft Computing* 18.2 (2014), pp. 393–412. ISSN: 1433-7479. DOI: 10.1007/s00500-013-1065-z.

[42] F. M. Bianchi et al. "An agent-based algorithm exploiting multiple local dissimilarities for clusters mining and knowledge discovery". In: *Soft Computing* 21.5 (2017), pp. 1347–1369. ISSN: 1433-7479. DOI: 10.1007/s00500-015-1876-1.

[43] F. M. Bianchi et al. "An interpretable graph-based image classifier". In: *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014, pp. 2339–2346. DOI: 10.1109/IJCNN.2014.6889601.

[44] F. M. Bianchi et al. "Granular Computing Techniques for Classification and Semantic Characterization of Structured Data". In: *Cognitive Computation* 8.3 (2016), pp. 442–461. ISSN: 1866-9964. DOI: 10.1007/s12559-015-9369-1.

[45] F. M. Bianchi et al. “Identifying user habits through data mining on call data records”. In: *Engineering Applications of Artificial Intelligence* 54 (2016), pp. 49–61. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2016.05.007.

[46] S. Boccaletti et al. “Complex networks: Structure and dynamics”. In: *Physics Reports* 424.4 (2006), pp. 175 –308. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2005.10.009.

[47] P. Boldi and S. Vigna. “Axioms for centrality”. In: *Internet Mathematics* 10.3-4 (2014), pp. 222–262. DOI: 10.1080/15427951.2013.865686.

[48] A. Bordbar et al. “Elucidating dynamic metabolic physiology through network integration of quantitative time-course metabolomics”. In: *Scientific reports* 7 (2017), p. 46249. DOI: 10.1038/srep46249.

[49] K. M. Borgwardt and H. P. Kriegel. “Shortest-path kernels on graphs”. In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. 2005, 8 pp.–. DOI: 10.1109/ICDM.2005.132.

[50] B. E. Boser, I. Guyon, and V. Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.

[51] P. S. Bradley, O. L. Mangasarian, and W. N. Street. “Clustering via Concave Minimization”. In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. NIPS'96. Denver, Colorado: MIT Press. Cambridge, MA, USA, 1996, pp. 368–374.

[52] U. Brandes. “On variants of shortest-path betweenness centrality and their generic computation”. In: *Social Networks* 30.2 (2008), pp. 136 –145. ISSN: 0378-8733. DOI: 10.1016/j.socnet.2007.11.001.

[53] D. J. Brenner, J. T. Staley, and N. R. Krieg. “Classification of Procaryotic Organisms and the Concept of Bacterial Speciation”. In: *Bergey's Manual of Systematics of Archaea and Bacteria* (2015), pp. 1–9. DOI: 10.1002/9781118960608.bm00006.

[54] C. Bron and J. Kerbosch. “Algorithm 457: Finding All Cliques of an Undirected Graph”. In: *Commun. ACM* 16.9 (1973), pp. 575–577. ISSN: 0001-0782. DOI: 10.1145/362342.362367.

[55] E. Bullmore and O. Sporns. “Complex brain networks: graph theoretical analysis of structural and functional systems”. In: *Nature Reviews Neuroscience* 10.3 (2009), pp. 186–198. DOI: 10.1038/nrn2575.

[56] H. Bunke. “Graph-Based Tools for Data Mining and Machine Learning”. In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by P. Perner and A. Rosenfeld. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 7–19. ISBN: 978-3-540-45065-8. DOI: 10.1007/3-540-45065-3_2.

[57] H. Bunke. “Graph matching: Theoretical foundations, algorithms, and applications”. In: *Proceedings of Vision Interface*. Montreal, Canada, 2000, pp. 82–88.

[58] H. Bunke. “On a relation between graph edit distance and maximum common subgraph”. In: *Pattern Recognition Letters* 18.8 (1997), pp. 689 –694. ISSN: 0167-8655. DOI: 10.1016/S0167-8655(97)00060-3.

[59] H. Bunke and G. Allermann. “Inexact graph matching for structural pattern recognition”. In: *Pattern Recognition Letters* 1.4 (1983), pp. 245 –253. ISSN: 0167-8655. DOI: 10.1016/0167-8655(83)90033-8.

[60] S. Butler. "Algebraic aspects of the normalized Laplacian". In: *Recent Trends in Combinatorics*. Ed. by A. Beveridge et al. Cham: Springer International Publishing, 2016, pp. 295–315. ISBN: 978-3-319-24298-9. DOI: 10.1007/978-3-319-24298-9_13.

[61] Z. Béni et al. "Bioactivity-Guided Isolation of Antimicrobial and Antioxidant Metabolites from the Mushroom Tapinella atrotomentosa". In: *Molecules* 23.5 (2018). ISSN: 1420-3049. DOI: 10.3390/molecules23051082.

[62] C. Böde et al. "Network analysis of protein dynamics". In: *FEBS Letters* 581.15 (2007), pp. 2776–2782. DOI: 10.1016/j.febslet.2007.05.021.

[63] G. Carlsson. "Topological pattern recognition for point cloud data". In: *Acta Numerica* 23 (2014), 289–368. DOI: 10.1017/S0962492914000051.

[64] G. Carlsson. "Topology and data". In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308. DOI: 10.1090/S0273-0979-09-01249-X.

[65] F. Cazals and C. Karande. "A note on the problem of reporting maximal cliques". In: *Theoretical Computer Science* 407.1 (2008), pp. 564 –568. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2008.05.010.

[66] C. Cellucci. *Rethinking logic: Logic in relation to mathematics, evolution, and method*. Springer Science & Business Media, 2013.

[67] C.-C. Chang and C.-J. Lin. "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), p. 27.

[68] A. Chaouachi et al. "Multiobjective Intelligent Energy Management for a Microgrid". In: *IEEE Transactions on Industrial Electronics* 60.4 (2013), pp. 1688–1699. DOI: 10.1109/TIE.2012.2188873.

[69] B. Chapman and J. Chang. "Biopython: Python Tools for Computational Biology". In: *SIGBIO Newsl.* 20.2 (2000), pp. 15–19. ISSN: 0163-5697. DOI: 10.1145/360262.360268.

[70] S. Chauhan, M. Girvan, and E. Ott. "Spectral properties of networks with community structure". In: *Phys. Rev. E* 80 (5 2009), p. 056114. DOI: 10.1103/PhysRevE.80.056114.

[71] F. Chazal et al. "Convergence rates for persistence diagram estimation in topological data analysis". In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 3603–3635.

[72] S. Chen, B. Ma, and K. Zhang. "On the similarity metric and the distance metric". In: *Theoretical Computer Science* 410.24 (2009). Formal Languages and Applications: A Collection of Papers in Honor of Sheng Yu, pp. 2365 –2376. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.02.023.

[73] Y. Chen, M. R. Gupta, and B. Recht. "Learning kernels from indefinite similarities". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 145–152. DOI: 10.1145/1553374.1553393.

[74] Y. Chen et al. "Similarity-based classification: Concepts and algorithms". In: *Journal of Machine Learning Research* 10.Mar (2009), pp. 747–776.

[75] F. R. K. Chung. *Spectral Graph Theory*. 2nd ed. Providence, USA: AMS, 1997. ISBN: 978-0-8218-0315-8.

[76] A. Cinti et al. "A Novel Algorithm for Online Inexact String Matching and its FPGA Implementation". In: *Cognitive Computation* (2019). ISSN: 1866-9964. DOI: 10.1007/s12559-019-09646-y.

[77] P. J. A. Cock et al. "Biopython: freely available Python tools for computational molecular biology and bioinformatics". In: *Bioinformatics* 25.11 (2009), pp. 1422–1423. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp163.

[78] R. Cohen et al. "Resilience of the Internet to Random Breakdowns". In: *Phys. Rev. Lett.* 85 (21 2000), pp. 4626–4628. DOI: 10.1103/PhysRevLett.85.4626.

[79] T. Cokelaer et al. "BioServices: a common Python package to access biological Web Services programmatically". In: *Bioinformatics* 29.24 (2013), pp. 3241–3242. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt547.

[80] M. Colafranceschi et al. "Structure-related statistical singularities along protein sequences: A correlation study". In: *Journal of chemical information and modeling* 45.1 (2005), pp. 183–189. DOI: 10.1021/ci049838m.

[81] M. Collins and N. Duffy. "Convolution Kernels for Natural Language". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS'01. Vancouver, British Columbia, Canada: MIT Press. Cambridge, MA, USA, 2001, pp. 625–632.

[82] A. Colorni, M. Dorigo, and V. Maniezzo. "Distributed optimization by ant colonies". In: *Toward a practice of autonomous systems: proceedings of the First European Conference on Artificial Life*. MIT Press. 1992, p. 134.

[83] A. Cornish-Bowden and M. L. Cárdenas. "Information transfer in metabolic pathways". In: *European Journal of Biochemistry* 268.24 (2001), pp. 6616–6624. DOI: 10.1046/j.0014-2956.2001.02616.x.

[84] A. Cornish-Bowden and M. L. Cárdenas. "Irreversible reactions in metabolic simulations: how reversible is irreversible". In: *Animating the cellular map*. Ed. by J.-H. S. Hofmeyr, H. M. Rohwer, and J. L. Snoep. Stellenbosch University Press, 2000, pp. 65–71.

[85] C. Cortes, M. Mohri, and A. Rostamizadeh. "Learning non-linear combinations of kernels". In: *Advances in neural information processing systems*. 2009, pp. 396–404.

[86] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[87] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[88] T. M. Cover. "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition". In: *IEEE transactions on electronic computers* EC-14.3 (1965), pp. 326–334. DOI: 10.1109/PGEC.1965.264137.

[89] T. F. Covões and E. R. Hruschka. "Towards improving cluster-based feature selection with a simplified silhouette filter". In: *Information Sciences* 181.18 (2011), pp. 3766–3782. ISSN: 0020-0255. DOI: 10.1016/j.ins.2011.04.050.

[90] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[91] P. Csermely, V. Agoston, and S. Pongor. "The efficiency of multi-target drugs: the network approach might help drug design". In: *Trends in pharmacological sciences* 26.4 (2005), pp. 178–182. DOI: 10.1016/j.tips.2005.02.007.

[92] P. Csermely et al. "Disordered proteins and network disorder in network descriptions of protein structure, dynamics and function: hypotheses and a comprehensive review". In: *Current Protein and Peptide Science* 13.1 (2012), pp. 19–33. DOI: 10.2174/138920312799277992.

[93] P. Csermely et al. "Structure and dynamics of core/periphery networks". In: *Journal of Complex Networks* 1.2 (2013), pp. 93–123. ISSN: 2051-1310. DOI: 10.1093/comnet/cnt016.

[94] P. Csermely et al. "Structure and dynamics of molecular networks: A novel paradigm of drug discovery: A comprehensive review". In: *Pharmacology & Therapeutics* 138.3 (2013), pp. 333 –408. ISSN: 0163-7258. DOI: 10.1016/j.pharmthera.2013.01.016.

[95] E. R. van Dam and W. H. Haemers. "Which graphs are determined by their spectrum?" In: *Linear Algebra and its Applications* 373 (2003). Combinatorial Matrix Theory Conference (Pohang, 2002), pp. 241 –272. ISSN: 0024-3795. DOI: 10.1016/S0024-3795(03)00483-X.

[96] N. Das et al. "Comparison of Different Graph Distance Metrics for Semantic Text Based Classification". en. In: *Polibits* (2014), pp. 51 –58. ISSN: 1870-9044.

[97] N. Das et al. "Using Graphs and Semantic Information to Improve Text Classifiers". In: *Advances in Natural Language Processing*. Ed. by A. Przepiórkowski and M. Ogrodniczuk. Cham: Springer International Publishing, 2014, pp. 324–336. ISBN: 978-3-319-10888-9.

[98] D. L. Davies and D. W. Bouldin. "A Cluster Separation Measure". In: *IEEE transactions on pattern analysis and machine intelligence* PAMI-1.2 (1979), pp. 224–227. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766909.

[99] E. De Santis, A. Rizzi, and A. Sadeghian. "A cluster-based dissimilarity learning approach for localized fault classification in Smart Grids". In: *Swarm and evolutionary computation* 39 (2018), pp. 267–278. DOI: 10.1016/j.swevo.2017.10.007.

[100] E. De Santis et al. "Dissimilarity Space Representations and Automatic Feature Selection for Protein Function Prediction". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489115.

[101] E. De Santis et al. "Evolutionary Optimization of an Affine Model for Vulnerability Characterization in Smart Grids". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489749.

[102] E. De Santis et al. "Modeling and recognition of smart grid faults by a combined approach of dissimilarity learning and one-class classification". In: *Neurocomputing* 170 (2015), pp. 368–383. DOI: 10.1016/j.neucom.2015.05.112.

[103] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[104] M. Dehmer, L. A. J. Mueller, and F. Emmert-Streib. "Quantitative Network Measures as Biomarkers for Classifying Prostate Cancer Disease States: A Systems Approach to Diagnostic Biomarkers". In: *PLOS ONE* 8.11 (2013), pp. 1–8. DOI: 10.1371/journal.pone.0077602.

[105] G. Del Vescovo and A. Rizzi. "Automatic classification of graphs by symbolic histograms". In: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. IEEE. 2007, pp. 410–416. DOI: 10.1109/GrC.2007.140.

[106] G. Del Vescovo and A. Rizzi. "Online Handwriting Recognition by the Symbolic Histograms Approach". In: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. 2007, pp. 686–686. DOI: 10.1109/GrC.2007.141.

[107] G. Del Vescovo et al. "On the problem of modeling structured data with the MinSOD representative". In: *International Journal of Computer Theory and Engineering* 6.1 (2014), p. 9. DOI: 10.7763/IJCTE.2014.V6.827.

[108] L. Dethlefsen, M. McFall-Ngai, and D. A. Relman. "An ecological and evolutionary perspective on human–microbe mutualism and disease". In: *Nature* 449.7164 (2007), p. 811. DOI: 10.1038/nature06245.

[109] A. Deutsch et al. "A query language for XML". In: *Computer Networks* 31.11 (1999), pp. 1155 –1169. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(99)00020-1.

[110] J. L. Devore and R. Peck. *Statistics : the exploration and analysis of data*. 4th ed. Pacific Grove, CA : Brooks/Cole, 2001. ISBN: 0534358675 (hbk.)

[111] A. Di Noia, P. Montanari, and A. Rizzi. "Occupational Diseases Risk Prediction by Cluster Analysis and Genetic Optimization". In: *Proceedings of the International Joint Conference on Computational Intelligence-Volume 1*. SCITEPRESS-Science and Technology Publications, Lda. 2014, pp. 68–75.

[112] A. Di Noia, P. Montanari, and A. Rizzi. "Occupational Diseases Risk Prediction by Genetic Optimization: Towards a Non-exclusive Classification Approach". In: *Computational Intelligence*. Springer, 2016, pp. 63–77.

[113] A. Di Noia et al. "Supervised machine learning techniques and genetic optimization for occupational diseases risk prediction". In: *Soft Computing* (2019). ISSN: 1433-7479. DOI: 10.1007/s00500-019-04200-2.

[114] L. Di Paola et al. "Protein Contact Networks: An Emerging Paradigm in Chemistry". In: *Chemical Reviews* 113.3 (2013), pp. 1598–1613. DOI: 10.1021/cr3002356.

[115] L. Di Paola and A. Giuliani. "Protein contact network topology: a natural language for allostery". In: *Current Opinion in Structural Biology* 31 (2015), pp. 43 –48. ISSN: 0959-440X. DOI: 10.1016/j.sbi.2015.03.001.

[116] L. Di Paola and A. Giuliani. "Protein–Protein Interactions: The Structural Foundation of Life Complexity". In: *eLS*. John Wiley & Sons, 2017, pp. 1–12. ISBN: 9780470015902. DOI: 10.1002/9780470015902.a0001346.pub2.

[117] D. V. Dibrova, M. Y. Galperin, and A. Y. Mulkidjanian. "Phylogenomic reconstruction of archaeal fatty acid metabolism". In: *Environmental Microbiology* 16.4 (), pp. 907–918. DOI: 10.1111/1462-2920.12359.

[118] H. Ding et al. "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures". In: *Proc. VLDB Endow.* 1.2 (2008), pp. 1542–1552. ISSN: 2150-8097. DOI: 10.14778/1454159.1454226.

[119] S. Ding, M. Du, and H. Zhu. "Survey on granularity clustering". In: *Cognitive neurodynamics* 9.6 (2015), pp. 561–572.

[120] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. "Critical phenomena in complex networks". In: *Rev. Mod. Phys.* 80 (4 2008), pp. 1275–1335. DOI: 10.1103/RevModPhys.80.1275.

[121] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[122] A. Duardo-Sánchez et al. "Modeling Complex Metabolic Reactions, Ecological Systems, and Financial and Legal Networks with MIANN Models Based on Markov-Wiener Node Descriptors". In: *Journal of Chemical Information and Modeling* 54.1 (2014), pp. 16–29. DOI: 10.1021/ci400280n.

[123] F. L. Duecker, F. Reuß, and P. Heretsch. "Rearranged ergostane-type natural products: chemistry, biology, and medicinal aspects". In: *Org. Biomol. Chem.* 17 (7 2019), pp. 1624–1633. DOI: 10.1039/C8OB02325E.

[124] M. Dufrêne and P. Legendre. "Species assemblages and indicator species: the need for a flexible asymmetrical approach". In: *Ecological monographs* 67.3 (1997), pp. 345–366. DOI: 10.2307/2963459.

[125] R. P. Duin and E. Pękalska. "The dissimilarity space: Bridging structural and statistical pattern recognition". In: *Pattern Recognition Letters* 33.7 (2012). Special Issue on Awards from ICPR 2010, pp. 826 –832. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2011.04.019.

[126] D. M. Eler et al. "Simplified Stress and Simplified Silhouette Coefficient to a Faster Quality Evaluation of Multidimensional Projection Techniques and Feature Spaces". In: *Information Visualisation (iV), 2015 19th International Conference on*. IEEE. 2015, pp. 133–139. DOI: 10.1109/iV.2015.33.

[127] D. Emms et al. "A matrix representation of graphs and its spectrum as a graph invariant". In: *the electronic journal of combinatorics* 13.1 (2006), p. 34.

[128] P. Erdős and A. Renyi. "On random graphs I". In: *Publicationes Mathematicae* 6 (1959), pp. 290–297.

[129] F. Escolano, E. R. Hancock, and M. A. Lozano. "Heat diffusion: Thermodynamic depth complexity of networks". In: *Phys. Rev. E* 85 (3 2012), p. 036206. DOI: 10.1103/PhysRevE.85.036206.

[130] M. Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Vol. 96. 34. 1996, pp. 226–231.

[131] E. Estrada. "Characterization of 3D molecular structure". In: *Chemical Physics Letters* 319.5-6 (2000), pp. 713–718. DOI: 10.1016/S0009-2614(00)00158-5.

[132] E. Estrada. "Universality in Protein Residue Networks". In: *Biophysical Journal* 98.5 (2010), pp. 890 –900. ISSN: 0006-3495. DOI: 10.1016/j.bpj.2009.11.017.

[133] E. Estrada and J. A. Rodriguez-Velazquez. "Complex networks as hypergraphs". In: *arXiv preprint physics/0505137* (2005).

[134] E. Estrada and J. A. Rodriguez-Velazquez. "Subgraph centrality in complex networks". In: *Physical Review E* 71.5 (2005), p. 056103. DOI: 10.1103/PhysRevE.71.056103.

[135] R.-E. Fan et al. "LIBLINEAR: A Library for Large Linear Classification". In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874.

[136] K. Faust and J. Raes. "Microbial interactions: from networks to models". In: *Nature Reviews Microbiology* 10.8 (2012), p. 538. DOI: 10.1038/nrmicro2832.

[137] T. Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[138] J. G. Ferry. "Acetate Metabolism in Anaerobes from the Domain Archaea". In: *Life* 5.2 (2015), pp. 1454–1471. ISSN: 2075-1729. DOI: 10.3390/life5021454.

[139] O. Fiehn and W. Weckwerth. "Deciphering metabolic networks". In: *European Journal of Biochemistry* 270.4 (), pp. 579–588. DOI: 10.1046/j.1432-1033.2003.03427.x.

[140] R. A. Fisher. "The statistical utilization of multiple measurements". In: *Annals of eugenics* 8.4 (1938), pp. 376–386.

[141] R. W. Floyd. "Algorithm 97: Shortest Path". In: *Commun. ACM* 5.6 (1962), pp. 345–. ISSN: 0001-0782. DOI: 10.1145/367766.368168.

[142] L. K. Gallos and N. H. Fefferman. "Revealing effective classifiers through network comparison". In: *EPL (Europhysics Letters)* 108.3 (2014), p. 38001. DOI: 10.1209/0295-5075/108/38001.

[143] J. Gao, B. Barzel, and A.-L. Barabási. "Universal resilience patterns in complex networks". In: *Nature* 530.7590 (2016), pp. 307–312. DOI: 10.1038/nature16948.

[144] A. Gardner et al. "On the Definiteness of Earth Mover's Distance and Its Relation to Set Intersection". In: *IEEE Transactions on Cybernetics* 48.11 (2018), pp. 3184–3196. ISSN: 2168-2267. DOI: 10.1109/TCYB.2017.2761798.

[145] T. Gärtner, P. Flach, and S. Wrobel. "On Graph Kernels: Hardness Results and Efficient Alternatives". In: *Learning Theory and Kernel Machines*. Ed. by B. Schölkopf and M. K. Warmuth. Springer, Berlin, Heidelberg, 2003, pp. 129–143. ISBN: 978-3-540-45167-9.

[146] T. Gaudelet, N. Malod-Dognin, and N. Pržulj. "Higher-order molecular organization as a source of biological function". In: *Bioinformatics* 34.17 (2018), pp. i944–i953. DOI: 10.1093/bioinformatics/bty570.

[147] S. Ghosh et al. "The journey of graph kernels through two decades". In: *Computer Science Review* 27 (2018), pp. 88 –111. ISSN: 1574-0137.

[148] R. Ghrist. *Elementary applied topology*. 1st ed. Seattle, USA: Createspace, 2014.

[149] A. Giuliani. "Some Notes on the Actual Status of Quantitative Approaches in Biotechnology". In: *Current Biotechnology* 7.6 (2018), pp. 406–408. ISSN: 2211-5501/2211-551X. DOI: 10.2174/2211550107061902121115 45.

[150] A. Giuliani, S. Filippi, and M. Bertolaso. "Why network approach can promote a new way of thinking in biology". In: *Frontiers in Genetics* 5 (2014), p. 83. ISSN: 1664-8021. DOI: 10.3389/fgene.2014.00083.

[151] C. Giusti, R. Ghrist, and D. S. Bassett. "Two's company, three (or more) is a simplex". In: *Journal of Computational Neuroscience* 41.1 (2016), pp. 1–14. ISSN: 1573-6873. DOI: 10.1007/s10827-016-0608-6.

[152] K.-I. Goh, B. Kahng, and D Kim. "Universal behavior of load distribution in scale-free networks". In: *Physical Review Letters* 87.27 (2001), p. 278701. DOI: 10.1103/PhysRevLett.87.278701.

[153] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Reading, Massachusetts, USA: Addison-Wesley, 1989.

[154] M. Gönen and E. Alpaydin. "Localized multiple kernel learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 352–359. DOI: 10.1145/1390156.1390201.

[155] M. Gönen and E. Alpaydın. "Multiple kernel learning algorithms". In: *Journal of machine learning research* 12.Jul (2011), pp. 2211–2268.

[156] H. González-Díaz and P. Riera-Fernández. "New Markov-Autocorrelation Indices for Re-evaluation of Links in Chemical and Biological Complex Networks used in Metabolomics, Parasitology, Neurosciences, and Epidemiology". In: *Journal of Chemical Information and Modeling* 52.12 (2012), pp. 3331–3340. DOI: 10.1021/ci300321f.

[157] J. L. Green, B. J. M. Bohannan, and R. J. Whitaker. "Microbial Biogeography: From Taxonomy to Traits". In: *Science* 320.5879 (2008), pp. 1039–1043. ISSN: 0036-8075. DOI: 10.1126/science.1153475.

[158] J. Gu, B. Hua, and S. Liu. "Spectral distances on graphs". In: *Discrete Applied Mathematics* 190-191 (2015), pp. 56 –74. ISSN: 0166-218X. DOI: 10.1016/j.dam.2015.04.011.

[159] E. Guarnera and I. N. Berezovsky. "Allosteric sites: remote control in regulation of protein activity". In: *Current Opinion in Structural Biology* 37 (2016), pp. 1 –8. ISSN: 0959-440X. DOI: 10.1016/j.sbi.2015.10.004.

[160] S. Guha, R. Rastogi, and K. Shim. "CURE: an efficient clustering algorithm for large databases". In: *ACM Sigmod Record*. Vol. 27. 2. ACM. 1998, pp. 73–84.

[161] R. Guimerà and L. A. N. Amaral. "Functional cartography of complex metabolic networks". In: *Nature* 433.7028 (2005), p. 895. DOI: 10.1038/nature03288.

[162] I. Gutman and B. Zhou. "Laplacian energy of a graph". In: *Linear Algebra and its Applications* 414.1 (2006), pp. 29 –37. ISSN: 0024-3795. DOI: 10.1016/j.laa.2005.09.008.

[163] B. Haasdonk. "Feature space interpretation of SVMs with indefinite kernels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (2005), pp. 482–492. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.78.

[164] R. W. Hamming. "Error detecting and error correcting codes". In: *Bell Labs Technical Journal* 29.2 (1950), pp. 147–160.

[165] L. Han et al. "Graph characterizations from von Neumann entropy". In: *Pattern Recognition Letters* 33.15 (2012), pp. 1958 –1967. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2012.03.016.

[166] Z. Harchaoui and F. Bach. "Image Classification with Segmentation Graph Kernels". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383049.

[167] D. R. Hofstadter. *I am a Strange Loop*. Basic Books, 2007.

[168] P. Holme and J. Saramäki. "Temporal networks". In: *Physics Reports* 519.3 (2012), pp. 97 –125. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2012.03.001.

[169] D. Horak, S. Maletić, and M. Rajković. "Persistent homology of complex networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2009.03 (2009), P03034. DOI: 10.1088/1742-5468/2009/03/p03034.

[170] J. Horgan. "From complexity to perplexity". In: *Scientific American* 272.6 (1995), pp. 104–109.

[171] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[172] N. Howard and H. Lieberman. "BrainSpace: Relating Neuroscience to Knowledge About Everyday Life". In: *Cognitive Computation* 6.1 (2014), pp. 35–44.

[173] E. R. Hruschka, R. J. Campello, and L. N. de Castro. "Evolving clusters in gene-expression data". In: *Information Sciences* 176.13 (2006), pp. 1898–1927. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2005.07.015.

[174] M. Hu, Y. Chen, and J. T.-Y. Kwok. "Building sparse multiple-kernel SVM classifiers". In: *IEEE Transactions on Neural Networks* 20.5 (2009), pp. 827–839. DOI: 10.1109/TNN.2009.2014229.

[175] T. S. Jaakkola and D. Haussler. "Exploiting Generative Models in Discriminative Classifiers". In: *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. Cambridge, MA, USA: MIT Press, 1999, pp. 487–493. ISBN: 0-262-11245-0.

[176] P. Jaccard. "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines". In: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37.140 (1901), pp. 241–272. DOI: 10.5169/seals-266440.

[177] P. Jaccard. "Etude de la distribution florale dans une portion des Alpes et du Jura". In: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37.142 (1901), pp. 547–579. DOI: 10.5169/seals-266450.

[178] P. Jaccard. "The distribution of the flora in the Alpine zone". In: *New Phytologist* 11.2 (1912), pp. 37–50. DOI: 10.1111/j.1469-8137.1912.tb05611.x.

[179] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data clustering: a review". In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.

[180] S. Jalan. "Spectral analysis of deformed random networks". In: *Phys. Rev. E* 80 (4 2009), p. 046101. DOI: 10.1103/PhysRevE.80.046101.

[181] S. Jalan and J. N. Bandyopadhyay. "Random matrix analysis of network Laplacians". In: *Physica A: Statistical Mechanics and its Applications* 387.2 (2008), pp. 667 –674. ISSN: 0378-4371. DOI: 10.1016/j.physa.2007.09.026.

[182] J. R. Jang. "ANFIS: adaptive-network-based fuzzy inference system". In: *IEEE Transactions on Systems, Man, and Cybernetics* 23.3 (1993), pp. 665–685. DOI: 10.1109/21.256541.

[183] X. Ji and J. Bailey. "An Efficient Technique for Mining Approximately Frequent Substring Patterns". In: *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. 2007, pp. 325–330. DOI: 10.1109/ICDMW.2007.121.

[184] G. Jurman, R. Visintainer, and C. Furlanello. "An introduction to spectral distances in networks". In: *Neural Nets WIRN10*. Ed. by B. Apolloni et al. Vol. 226. Frontiers in Artificial Intelligence and Applications. IOS Press, 2011, pp. 227–234. DOI: 10.3233/978-1-60750-692-8-227.

[185] M. Kafri et al. "The Cost of Protein Production". In: *Cell Reports* 14.1 (2016), pp. 22 –31. ISSN: 2211-1247. DOI: 10.1016/j.celrep.2015.12.015.

[186] M. Kanehisa and S. Goto. "KEGG: kyoto encyclopedia of genes and genomes". In: *Nucleic acids research* 28.1 (2000), pp. 27–30.

[187] M. Kanehisa et al. "KEGG as a reference resource for gene and protein annotation". In: *Nucleic acids research* 44.D1 (2015), pp. D457–D462.

[188] M. Kanehisa et al. "KEGG: new perspectives on genomes, pathways, diseases and drugs". In: *Nucleic acids research* 45.D1 (2016), pp. D353–D361.

[189] K. Kanno et al. "Interacting Proteins Dictate Function of the Minimal START Domain Phosphatidylcholine Transfer Protein/StarD2". In: *Journal of Biological Chemistry* 282.42 (2007), pp. 30728–30736. DOI: 10.1074/jbc.M703745200.

[190] L. Katz. "A new status index derived from sociometric analysis". In: *Psychometrika* 18.1 (1953), pp. 39–43. DOI: 10.1007/BF02289026.

[191] L. Kaufman and P. Rousseeuw. "Clustering by means of Medoids". In: *Statistical Data Analysis Based on the L1 Norm and Related Methods*. Ed. by Y. Dodge. North Holland, 1987, pp. 405–416.

[192] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990. DOI: 10.1002/9780470316801.

[193] J. C. Kendrew et al. "A three-dimensional model of the myoglobin molecule obtained by x-ray analysis". In: *Nature* 181.4610 (1958), pp. 662–666.

[194] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[195] K. Kersting et al. *Benchmark Data Sets for Graph Kernels*. http://graphkernels.cs.tu-dortmund.de. 2016.

[196] S. S. Khan and M. G. Madden. "A Survey of Recent Trends in One Class Classification". In: *Artificial Intelligence and Cognitive Science*. Ed. by L. Coyle and J. Freyne. Vol. 6206. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 188–197. ISBN: 978-3-642-17079-9. DOI: 10.1007/978-3-642-17080-5_21.

[197] J. Kim and T. Wilhelm. "What is a complex graph?" In: *Physica A: Statistical Mechanics and its Applications* 387.11 (2008), pp. 2637 –2652. ISSN: 0378-4371. DOI: 10.1016/j.physa.2008.01.015.

[198] J. Kirchmair et al. "Predicting drug metabolism: experiment and/or computation?" In: *Nature Reviews Drug discovery* 14.6 (2015), pp. 387–404. DOI: 10.1038/nrd4581.

[199] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. "Optimization by simulated annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[200] A. H. Knoll. "Paleobiological perspectives on early eukaryotic evolution". In: *Cold Spring Harbor Perspectives in Biology* 6.1 (2014), a016121. DOI: 10.1101/cshperspect.a016121.

[201] R. Kohavi. "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc. San Francisco, USA, 1995, pp. 1137–1143. ISBN: 1-55860-363-8.

[202] R. I. Kondor and J. Lafferty. "Diffusion kernels on graphs and other discrete structures". In: *In Proceedings of the ICML*. 2002, pp. 315–322.

[203] R. I. Kondor and J. Lafferty. "Diffusion kernels on graphs and other discrete structures". In: *Proceedings of the 19th international conference on machine learning*. Vol. 2002. 2002, pp. 315–322.

[204] A. Krishnan et al. "Proteins as networks: usefulness of graph theory in protein science". In: *Current Protein and Peptide Science* 9.1 (2008), pp. 28–38. DOI: 10.2174/138920308783565705.

[205] R. Kühn and J. van Mourik. "Spectra of modular and small-world matrices". In: *Journal of Physics A: Mathematical and Theoretical* 44.16 (2011), p. 165205. DOI: 10.1088/1751-8113/44/16/165205.

[206] V. Lacroix et al. "An Introduction to Metabolic Networks and Their Structural Analysis". In: *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 5.4 (2008), pp. 594–617. ISSN: 1545-5963. DOI: 10.1109/TCBB.2008.79.

[207] C. H. Lampert. "Kernel Methods in Computer Vision". In: *Foundations and Trends® in Computer Graphics and Vision* 4.3 (2009), pp. 193–285. ISSN: 1572-2740. DOI: 10.1561/0600000027.

[208] G. R. Lanckriet et al. "Learning the kernel matrix with semidefinite programming". In: *Journal of Machine learning research* 5.Jan (2004), pp. 27–72.

[209] S. de Lange, M. de Reus, and M. Van Den Heuvel. "The Laplacian spectrum of neural networks". In: *Frontiers in Computational Neuroscience* 7 (2014), p. 189. ISSN: 1662-5188. DOI: 10.3389/fncom.2013.00189.

[210] J. Laub and K.-R. Müller. "Feature Discovery in Non-Metric Pairwise Data". In: *J. Mach. Learn. Res.* 5 (2004), pp. 801–818. ISSN: 1532-4435.

[211] R. B. Laughlin et al. "The middle way". In: *Proceedings of the National Academy of Sciences* 97.1 (2000), pp. 32–37. ISSN: 0027-8424. DOI: 10.1073/pnas.97.1.32.

[212] D.-S. Lee et al. "The implications of human metabolic network topology for disease comorbidity". In: *Proceedings of the National Academy of Sciences* 105.29 (2008), pp. 9880–9885. ISSN: 0027-8424. DOI: 10.1073/pnas.0802208105.

[213] S. Leonori et al. "An optimized microgrid energy management system based on FIS-MO-GA paradigm". In: *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2017, pp. 1–6. DOI: 10.1109/FUZZ-IEEE.2017.8015438.

[214] S. Leonori et al. "Multi objective optimization of a fuzzy logic controller for energy management in microgrids". In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. 2016, pp. 319–326. DOI: 10.1109/CEC.2016.7743811.

[215] S. Leonori et al. "Optimization of a microgrid energy management system based on a Fuzzy Logic Controller". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 2016, pp. 6615–6620. DOI: 10.1109/IECON.2016.7793965.

[216] S. Leonori et al. "ANFIS Microgrid Energy Management System Synthesis by Hyperplane Clustering Supported by Neurofuzzy Min–Max Classifier". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 3.3 (2019), pp. 193–204. ISSN: 2471-285X. DOI: 10.1109/TETCI.2018.2880815.

[217] S. Leonori et al. "ANFIS Synthesis by Clustering for Microgrids EMS Design". In: *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,* INSTICC. SciTePress, 2017, pp. 328–337. ISBN: 978-989-758-274-5. DOI: 10.5220/0006514903280337.

[218] S. Leonori et al. "Microgrid Energy Management Systems Design by Computational Intelligence Techniques". In: *IEEE transactions on information theory* (2020). Under Review.

[219] C. Leslie, E. Eskin, and W. Stafford Noble. "The Spectrum Kernel: A String Kernel for SVM Protein Classification". In: *Pacific Symposium on Biocomputing*. Vol. 7. World Scientific, 2002, pp. 564–575. DOI: 10.1142/9789812799623_0053.

[220] C. S. Leslie et al. "Mismatch string kernels for discriminative protein classification". In: *Bioinformatics* 20.4 (2004), pp. 467–476. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg431.

[221] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady* 10.8 (1966), pp. 707–710.

[222] D. P. Lewis, T. Jebara, and W. S. Noble. "Nonstationary kernel combination". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 553–560. DOI: 10.1145/1143844.1143914.

[223] J.-P. Li et al. "A Species Conserving Genetic Algorithm for Multimodal Function Optimization". In: *Evolutionary Computation* 10.3 (2002), pp. 207–234. DOI: 10.1162/106365602760234081.

[224] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[225] T. Y. Lin, Y. Y. Yao, and L. A. Zadeh. *Data mining, rough sets and granular computing*. Vol. 95. Physica, 2013.

[226] L. Livi, A. Giuliani, and A. Rizzi. "Toward a multilevel representation of protein molecules: Comparative approaches to the aggregation/folding propensity problem". In: *Information Sciences* 326 (2016), pp. 134–145. ISSN: 0020-0255. DOI: 10.1016/j.ins.2015.07.043.

[227] L. Livi, A. Giuliani, and A. Sadeghian. "Characterization of graphs for protein structure modeling and recognition of solubility". In: *Current Bioinformatics* 11.1 (2016), pp. 106–114. DOI: 10.2174/1574893611666151109175216.

[228] L. Livi and A. Rizzi. "Graph ambiguity". In: *Fuzzy Sets and Systems* 221 (2013), pp. 24 –47. ISSN: 0165-0114. DOI: 10.1016/j.fss.2013.01.001.

[229] L. Livi and A. Rizzi. "The graph matching problem". In: *Pattern Analysis and Applications* 16.3 (2013), pp. 253–283. ISSN: 1433-755X. DOI: 10.1007/s10044-012-0284-8.

[230] L. Livi, A. Rizzi, and A. Sadeghian. "Granular modeling and computing approaches for intelligent analysis of non-geometric data". In: *Applied Soft Computing* 27 (2015), pp. 567 –574. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2014.08.072.

[231] L. Livi, A. Rizzi, and A. Sadeghian. "Optimized dissimilarity space embedding for labeled graphs". In: *Information Sciences* 266 (2014), pp. 47–64. DOI: 10.1016/j.ins.2014.01.005.

[232] L. Livi and A. Sadeghian. "Granular computing, computational intelligence, and the analysis of non-geometric input spaces". In: *Granular Computing* 1.1 (2016), pp. 13–20. ISSN: 2364-4974. DOI: 10.1007/s41066-015-0003-0.

[233] L. Livi et al. "A generative model for protein contact networks". In: *Journal of Biomolecular Structure and Dynamics* 34.7 (2016), pp. 1441–1454. DOI: 10.1080/07391102.2015.1077736.

[234] L. Livi et al. "Analysis of heat kernel highlights the strongly modular and heat-preserving structure of proteins". In: *Physica A: Statistical Mechanics and its Applications* 441 (2016), pp. 199 –214. ISSN: 0378-4371. DOI: 10.1016/j.physa.2015.08.059.

[235] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489.

[236] M. Y. Lobanov, N. Bogatyreva, and O. Galzitskaya. "Radius of gyration as an indicator of protein structure compactness". In: *Molecular Biology* 42.4 (2008), pp. 623–628. DOI: 10.1134/S0026893308040195.

[237] H. Lodish et al. *Molecular Cell Biology*. 4th ed. New York, USA: W. H. Freeman, 2000. ISBN: 0-7167-3136-3.

[238] U. von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416. ISSN: 1573-1375. DOI: 10.1007/s11222-007-9033-z.

[239] J. Ma and X. Ma. "A review of forecasting algorithms and energy management strategies for microgrids". In: *Systems Science & Control Engineering* 6.1 (2018), pp. 237–248. DOI: 10.1080/21642583.2018.1480979.

[240] Y. Ma et al. "Segmentation of Multivariate Mixed Data via Lossy Data Coding and Compression". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.9 (2007), pp. 1546–1562. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1085.

[241] R. W. A. Mackenzie and B. T. Elliott. "Akt/PKB activation and insulin signaling: a novel insulin signaling pathway in the treatment of type 2 diabetes". In: *Diabetes, metabolic syndrome and obesity: targets and therapy* 7 (2014), p. 55. DOI: 10.2147/DMSO.S48260.

[242] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Oakland, USA, 1967, pp. 281–297.

[243] E. Maiorino et al. "Noise Sensitivity of an Information Granules Filtering Procedure by Genetic Optimization for Inexact Sequential Pattern Mining". In: *Computational Intelligence*. Ed. by J. J. Merelo et al. Cham: Springer International Publishing, 2016, pp. 131–150. ISBN: 978-3-319-26393-9. DOI: 10.1007/978-3-319-26393-9_9.

[244] E. Maiorino et al. "Spectral reconstruction of protein contact networks". In: *Physica A: Statistical Mechanics and its Applications* 471 (2017), pp. 804–817. DOI: 10.1016/j.physa.2016.12.046.

[245] P. W. Majerus, M. V. Kisseleva, and F. A. Norris. "The role of phosphatases in inositol signaling reactions". In: *Journal of Biological Chemistry* 274.16 (1999), pp. 10669–10672. DOI: 10.1074/jbc.274.16.10669.

[246] N. Malod-Dognin and N. Pržulj. "Functional geometry of protein interactomes". In: *Bioinformatics* (2019). ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz146.

[247] J. R. Marchesi et al. "The gut microbiota and host health: a new clinical frontier". In: *Gut* 65.2 (2016), pp. 330–339. ISSN: 0017-5749. DOI: 10.1136/gutjnl-2015-309990.

[248] W. Martin. "Archaebacteria (Archaea) and the origin of the eukaryotic nucleus". In: *Current Opinion in Microbiology* 8.6 (2005), pp. 630–637. ISSN: 1369-5274. DOI: 10.1016/j.mib.2005.10.004.

[249] A. Martino, A. Rizzi, and F. M. Frattale Mascioli. "Supervised Approaches for Protein Function Prediction by Topological Data Analysis". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489307.

[250] A. Martino, F. M. Frattale Mascioli, and A. Rizzi. "On the Optimisation of Embedding Spaces via Information Granulation for Pattern Recognition". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Under Review. 2020.

[251] A. Martino, A. Giuliani, and A. Rizzi. "Granular Computing Techniques for Bioinformatics Pattern Recognition Problems in Non-metric Spaces". In: *Computational Intelligence for Pattern Recognition*. Ed. by W. Pedrycz and S.-M. Chen. Cham: Springer International Publishing, 2018, pp. 53–81. ISBN: 978-3-319-89629-8. DOI: 10.1007/978-3-319-89629-8_3.

[252] A. Martino, A. Giuliani, and A. Rizzi. "(Hyper)Graph Embedding and Classification via Simplicial Complexes". In: *Algorithms* 12.11 (2019). ISSN: 1999-4893. DOI: 10.3390/a12110223.

[253] A. Martino, A. Giuliani, and A. Rizzi. "The Universal Phenotype". In: *Organisms. Journal of Biological Sciences* 3.2 (2019), pp. 8–10.

[254] A. Martino and A. Rizzi. "(Hyper)Graph Kernels over Simplicial Complexes". In: *Pattern Recognition* (2019). Under Review.

[255] A. Martino, A. Rizzi, and F. M. Frattale Mascioli. "Distance Matrix Pre-Caching and Distributed Computation of Internal Validation Indices in k-medoids Clustering". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489101.

[256] A. Martino, A. Rizzi, and F. M. Frattale Mascioli. "Efficient Approaches for Solving the Large-Scale k-medoids Problem". In: *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI*, INSTICC. SciTePress, 2017, pp. 338–347. ISBN: 978-989-758-274-5. DOI: 10.5220/0006515003380347.

[257] A. Martino, A. Rizzi, and F. M. Frattale Mascioli. "Efficient Approaches for Solving the Large-Scale k-Medoids Problem: Towards Structured Data". In: *Computational Intelligence: 9th International Joint Conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1-3, 2017 Revised Selected Papers*. Ed. by C. Sabourin et al. Cham: Springer International Publishing, 2019, pp. 199–219. ISBN: 978-3-030-16469-0. DOI: 10.1007/978-3-030-16469-0_11.

[258] A. Martino et al. "Calibration Techniques for Binary Classification Problems: A Comparative Analysis". In: *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: NCTA, (IJCCI 2019)*. INSTICC. SciTePress, 2019, pp. 487–495. ISBN: 978-989-758-384-1. DOI: 10.5220/0008165504870495.

[259] A. Martino et al. "Data Mining by Evolving Agents for Clusters Discovery and Metric Learning". In: *Neural Advances in Processing Nonlinear Dynamic Signals*. Ed. by A. Esposito et al. Cham: Springer International Publishing, 2019, pp. 23–35. ISBN: 978-3-319-95098-3. DOI: 10.1007/978-3-319-95098-3_3.

[260] A. Martino et al. "Metabolic networks classification and knowledge discovery by information granulation". In: *Computational Biology and Chemistry* 84 (2020), p. 107187. ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2019.107187.

[261] A. Martino et al. "Modelling and Recognition of Protein Contact Networks by Multiple Kernel Learning and Dissimilarity Representations". In: *Information Sciences* (2019). Under Review.

[262] A. Martino et al. “Supervised Approaches for Function Prediction of Proteins Contact Networks from Topological Structure Information”. In: *Image Analysis: 20th Scandinavian Conference, SCIA 2017, Tromsø, Norway, June 12–14, 2017, Proceedings, Part I*. Ed. by P. Sharma and F. M. Bianchi. Cham: Springer International Publishing, 2017, pp. 285–296. DOI: 10.1007/978-3-319-59126-1_24.

[263] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[264] P. McDonald and R. Meyers. “Diffusions on graphs, Poisson problems and spectral geometry”. In: *Transactions of the American Mathematical Society* 354.12 (2002), pp. 5111–5136.

[265] P. N. McGraw and M. Menzinger. “Laplacian spectra as a diagnostic tool for network structure and dynamics”. In: *Phys. Rev. E* 77 (3 2008), p. 031102. DOI: 10.1103/PhysRevE.77.031102.

[266] P. J. McMurdie and S. Holmes. “phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data”. In: *PLOS ONE* 8.4 (2013), pp. 1–11. DOI: 10.1371/journal.pone.0061217.

[267] M. L. Mehta. *Random matrices*. 3rd ed. Vol. 142. Academic Press, Elsevier, 2004.

[268] J. M. Mendel. “Fuzzy logic systems for engineering: a tutorial”. In: *Proceedings of the IEEE* 83.3 (1995), pp. 345–377. ISSN: 0018-9219. DOI: 10.1109/5.364485.

[269] J. A. Méndez-Bermúdez et al. “Universality in the spectral and eigenfunction properties of random networks”. In: *Phys. Rev. E* 91 (3 2015), p. 032122. DOI: 10.1103/PhysRevE.91.032122.

[270] X. Meng et al. “Mllib: Machine learning in apache spark”. In: *Journal of Machine Learning Research* 17.34 (2016), pp. 1–7.

[271] J. Mercer. “Functions of positive and negative type, and their connection with the theory of integral equations”. In: *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character* 209 (1909), pp. 415–446.

[272] R. Merris. “Laplacian matrices of graphs: a survey”. In: *Linear Algebra and its Applications* 197-198 (1994), pp. 143 –176. ISSN: 0024-3795. DOI: 10.1016/0024-3795(94)90486-3.

[273] D. C. Mikulecky. “Network thermodynamics and complexity: a transition to relational systems theory”. In: *Computers & Chemistry* 25.4 (2001), pp. 369 – 391. ISSN: 0097-8485. DOI: 10.1016/S0097-8485(01)00072-9.

[274] S. Milgram. “The small world problem”. In: *Psychology today* 2.1 (1967), pp. 60–67.

[275] A. Mirshahvalad et al. “Dynamics of interacting information waves in networks”. In: *Phys. Rev. E* 89 (1 2014), p. 012809. DOI: 10.1103/PhysRevE.89.012809.

[276] T. M. Mitchell. *Machine Learning*. Boston, USA: McGraw-Hill, 1997.

[277] M. Mitrović and B. Tadić. “Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities”. In: *Phys. Rev. E* 80 (2 2009), p. 026123. DOI: 10.1103/PhysRevE.80.026123.

[278] B. Mohar. “Laplace eigenvalues of graphs—a survey”. In: *Discrete Mathematics* 109.1 (1992), pp. 171 –183. ISSN: 0012-365X. DOI: 10.1016/0012-365X(92)90288-Q.

[279] E. Mones, L. Vicsek, and T. Vicsek. "Hierarchy measure for complex networks". In: *PloS one* 7.3 (2012), e33799. DOI: 10.1371/journal.pone.0033799.

[280] J. W. Moon and L. Moser. "On cliques in graphs". In: *Israel Journal of Mathematics* 3.1 (1965), pp. 23–28. ISSN: 1565-8511. DOI: 10.1007/BF02760024.

[281] J. R. Munkres. *Elements of algebraic topology*. California, USA: Addison-Wesley, 1984.

[282] K. Murata and M. Wolf. "Cryo-electron microscopy for structural analysis of dynamic biological macromolecules". In: *Biochimica et Biophysica Acta (BBA) - General Subjects* 1862.2 (2018). SI: Biophysical Exploration of Dynamical Ordering of Biomolecular Systems, pp. 324 –334. ISSN: 0304-4165. DOI: 10.1016/j.bbagen.2017.07.020.

[283] R. Nader et al. "On the positive semi-definite property of similarity matrices". In: *Theoretical Computer Science* 755 (2019), pp. 13–28. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2018.06.052.

[284] D. Nayar and N. F. A. van der Vegt. "Cosolvent effects on polymer hydration drive hydrophobic collapse". In: *The Journal of Physical Chemistry B* 122.13 (2018), pp. 3587–3595. DOI: 10.1021/acs.jpcb.7b10780.

[285] C. F. A. Negre et al. "Eigenvector centrality for characterization of protein allosteric pathways". In: *Proceedings of the National Academy of Sciences* 115.52 (2018), E12201–E12208. ISSN: 0027-8424. DOI: 10.1073/pnas.1810452115.

[286] M. Neuhaus and H. Bunke. *Bridging the gap between graph edit distance and kernel machines*. Vol. 68. World Scientific, 2007.

[287] M. Neuhaus and H. Bunke. "Edit distance-based kernel functions for structural pattern classification". In: *Pattern Recognition* 39.10 (2006), pp. 1852–1863. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.04.012.

[288] M. Neumann et al. "Propagation kernels: efficient graph kernels from propagated information". In: *Machine Learning* 102.2 (2016), pp. 209–245. ISSN: 1573-0565. DOI: 10.1007/s10994-015-5517-9.

[289] M. E. J. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[290] M. E. J. Newman. "Power laws, Pareto distributions and Zipf's law". In: *Contemporary Physics* 46.5 (2005), pp. 323–351. DOI: 10.1080/00107510500052444.

[291] R. T. Ng and J. Han. "CLARANS: a method for clustering objects for spatial data mining". In: *IEEE Transactions on Knowledge and Data Engineering* 14.5 (2002), pp. 1003–1016. DOI: 10.1109/TKDE.2002.1033770.

[292] V. Nicosia, M. D. Domenico, and V. Latora. "Characteristic exponents of complex networks". In: *EPL (Europhysics Letters)* 106.5 (2014), p. 58005. DOI: 10.1209/0295-5075/106/58005.

[293] T. Niwa et al. "Bimodal protein solubility distribution revealed by an aggregation analysis of the entire ensemble of Escherichia coli proteins". In: *Proceedings of the National Academy of Sciences* 106.11 (2009), pp. 4201–4206. ISSN: 0027-8424. DOI: 10.1073/pnas.0811922106.

[294] H. F. Olivares-Rubio and A. Vega-López. "Fatty acid metabolism in fish species as a biomarker for environmental monitoring". In: *Environmental Pollution* 218 (2016), pp. 297 –312. ISSN: 0269-7491. DOI: 10.1016/j.envpol.2016.07.005.

[295] S. H. de Oliveira and C. M. Deane. "Exploring Folding Features in Protein Structure Prediction". In: *Biophysical Journal* 114.3, Supplement 1 (2018), 36a. ISSN: 0006-3495. DOI: 10.1016/j.bpj.2017.11.244.

[296] C. S. Ong et al. "Learning with Non-positive Kernels". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM. New York, NY, USA, 2004, pp. 81–. ISBN: 1-58113-838-5. DOI: 10.1145/1015330.1015443.

[297] J.-P. Onnela et al. "Taxonomies of networks from community structure". In: *Phys. Rev. E* 86 (3 2012), p. 036104. DOI: 10.1103/PhysRevE.86.036104.

[298] M. Orozco. "A theoretical view of protein dynamics". In: *Chem. Soc. Rev.* 43 (14 2014), pp. 5051–5066. DOI: 10.1039/C3CS60474H.

[299] M. C. Palumbo et al. "Functional essentiality from topology features in metabolic networks: A case study in yeast". In: *FEBS Letters* 579.21 (2005), pp. 4642–4646. ISSN: 0014-5793. DOI: 10.1016/j.febslet.2005.07.033.

[300] M. Panella et al. "ANFIS synthesis by hyperplane clustering". In: *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*. Vol. 1. 2001, 340–345 vol.1. DOI: 10.1109/NAFIPS.2001.944275.

[301] H.-S. Park and C.-H. Jun. "A simple and fast algorithm for K-medoids clustering". In: *Expert Systems with Applications* 36.2, Part 2 (2009), pp. 3336 –3341. ISSN: 0957–4174. DOI: 10.1016/j.eswa.2008.01.039.

[302] D. H. Parks et al. "A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life". In: *Nature Biotechnology* 36 (2018), pp. 996–1004. DOI: 10.1038/nbt.4229.

[303] E. Parzen. "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[304] L. Pauling, R. B. Corey, and H. R. Branson. "The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain". In: *Proceedings of the National Academy of Sciences* 37.4 (1951), pp. 205–211. DOI: 10.1073/pnas.37.4.205.

[305] W. Pedrycz. "Granular computing: an introduction". In: *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*. Vol. 3. IEEE. 2001, pp. 1349–1354. DOI: 10.1109/NAFIPS.2001.943745.

[306] W. Pedrycz. *Granular computing: analysis and design of intelligent systems*. CRC press, 2016.

[307] W. Pedrycz. "Human centricity in computing with fuzzy sets: an interpretability quest for higher order granular constructs". In: *Journal of Ambient Intelligence and Humanized Computing* 1.1 (2010), pp. 65–74.

[308] T. P. Peixoto. "Eigenvalue Spectra of Modular Networks". In: *Phys. Rev. Lett.* 111 (9 2013), p. 098701. DOI: 10.1103/PhysRevLett.111.098701.

[309] E. Pękalska and R. P. Duin. *The dissimilarity representation for pattern recognition: foundations and applications*. World Scientific, 2005. DOI: 10.1142/5965.

[310] E. Pękalska, R. P. Duin, and P. Paclík. "Prototype selection for dissimilarity-based classifiers". In: *Pattern Recognition* 39.2 (2006), pp. 189–208. DOI: 10.1016/j.patcog.2005.06.012.

[311] E. Pękalska et al. "Non-Euclidean or Non-metric Measures Can Be Informative". In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by D.-Y. Yeung et al. Springer, Berlin, Heidelberg, 2006, pp. 871–880. ISBN: 978-3-540-37241-7.

[312] J. B. Pereira et al. "Disrupted network topology in patients with stable and progressive mild cognitive impairment and Alzheimer's disease". In: *Cerebral Cortex* 26.8 (2016), pp. 3476–3493. ISSN: 1047-3211. DOI: 10.1093/cercor/bhw128.

[313] C. A. Peña-Reyes. "Evolutionary Fuzzy Modeling Human Diagnostic Decisions". In: *Annals of the New York Academy of Sciences* 1020.1 (2004), pp. 190–211. DOI: 10.1196/annals.1310.017.

[314] M. A. F. Pimentel et al. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215–249. DOI: 10.1016/j.sigpro.2013.12.026.

[315] F. Possemato and A. Rizzi. "Automatic text categorization by a Granular Computing approach: Facing unbalanced data sets". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2013, pp. 1–8. DOI: 10.1109/IJCNN.2013.6707082.

[316] D. M. W. Powers. "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.

[317] E. Ramadan, A. Tarafdar, and A. Pothen. "A hypergraph model for the yeast protein complex network". In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* 2004, pp. 189–. DOI: 10.1109/IPDPS.2004.1303205.

[318] S. Raschka. "BioPandas: Working with molecular structures in pandas DataFrames". In: *The Journal of Open Source Software* 2.14 (2017). DOI: 10.21105/joss.00279.

[319] P. Riera-Fernández et al. "New Markov–Shannon Entropy models to assess connectivity quality in complex networks: From molecular to cellular pathway, Parasite–Host, Neural, Industry, and Legal–Social networks". In: *Journal of Theoretical Biology* 293 (2012), pp. 174 –188. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2011.10.016.

[320] K. Riesen and H. Bunke. "Graph Classification by Means of Lipschitz Embedding". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6 (2009), pp. 1472–1483. DOI: 10.1109/TSMCB.2009.2019264.

[321] A. Rizzi. "Automatic Training of Min-Max Classifiers". In: *Neuro-Fuzzy Pattern Recognition*. Ed. by H. Bunke and A. Kandel, pp. 101–124. DOI: 10.1142/9789812792204_0005.

[322] A. Rizzi, M. Panella, and F. M. Frattale Mascioli. "Adaptive resolution min-max classifiers". In: *IEEE Transactions on Neural Networks* 13.2 (2002), pp. 402–414. DOI: 10.1109/72.991426.

[323] A. Rizzi et al. "A dissimilarity-based classifier for generalized sequences by a granular computing approach". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2013, pp. 1–8. DOI: 10.1109/IJCNN.2013.6707041.

[324] A. Rizzi et al. "A new Granular Computing approach for sequences representation and classification". In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2012, pp. 1–8. DOI: 10.1109/IJCNN.2012.6252680.

[325] A. Rizzi et al. "A recursive algorithm for fuzzy min-max networks". In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 6. 2000, 541–546 vol.6. DOI: 10.1109/IJCNN.2000.859451.

[326] M. F. Roberts. "Inositol in Bacteria and Archaea". In: *Biology of Inositols and Phosphoinositides: Subcellular Biochemistry*. Ed. by A. L. Majumder and B. B. Biswas. Boston, MA: Springer US, 2006, pp. 103–133. ISBN: 978-0-387-27600-7. DOI: 10.1007/0-387-27600-9_5.

[327] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[328] R. Ross et al. "Reduction in Obesity and Related Comorbid Conditions after Diet-Induced Weight Loss or Exercise-Induced Weight Loss in Men: A Randomized, Controlled Trial". In: *Annals of Internal Medicine* 133.2 (2000), pp. 92–103. ISSN: 0003-4819. DOI: 10.7326/0003-4819-133-2-200007180-00008.

[329] L. Rossi et al. "Characterizing graph symmetries through quantum Jensen-Shannon divergence". In: *Phys. Rev. E* 88 (3 2013), p. 032806. DOI: 10.1103/PhysRevE.88.032806.

[330] M. Rosvall et al. "Memory in network flows and its effects on spreading dynamics and community detection". In: *Nature communications* 5 (2014), p. 4630. DOI: 10.1038/ncomms5630.

[331] P. J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53 –65. ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7.

[332] M. Ružička. "Anwendung mathematisch-statistischer Methoden in der Geobotanik (Synthetische Bearbeitung von Aufnahmen)". In: *Biologia (Bratislava)* 13 (1958), pp. 647–661.

[333] J. Saramäki et al. "Generalizations of the clustering coefficient to weighted complex networks". In: *Physical Review E* 75.2 (2007), p. 027105. DOI: 10.1103/PhysRevE.75.027105.

[334] T. Schäfer, M. Selig, and P. Schönheit. "Acetyl-CoA synthetase (ADP forming) in archaea, a novel enzyme involved in acetate formation and ATP synthesis". In: *Archives of Microbiology* 159.1 (1993), pp. 72–83. ISSN: 1432-072X. DOI: 10.1007/BF00244267.

[335] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[336] B. Schölkopf et al. "New support vector algorithms". In: *Neural computation* 12.5 (2000), pp. 1207–1245. DOI: 10.1162/089976600300015565.

[337] S. Schuster, D. A. Fell, and T. Dandekar. "A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks". In: *Nature biotechnology* 18.3 (2000), pp. 326–332. DOI: 10.1038/73786.

[338] D. W. Scott. "On optimal and data-based histograms". In: *Biometrika* 66.3 (1979), pp. 605–610. ISSN: 0006-3444. DOI: 10.1093/biomet/66.3.605.

[339] R. Sedgewick. *Algorithms in C, Part 5: Graph Algorithms*. 3rd ed. Addison Wesley Professional, 2001.

[340] R. Seising. "Warren Weaver's "Science and complexity" Revisited". In: *Soft Computing in Humanities and Social Sciences*. Ed. by R. Seising and V. Sanz González. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 55–87. ISBN: 978-3-642-24672-2. DOI: 10.1007/978-3-642-24672-2_3.

[341] N. Shervashidze and K. Borgwardt. "Fast subtree kernels on graphs". In: *Advances in neural information processing systems*. 2009, pp. 1660–1668.

[342] N. Shervashidze et al. "Efficient graphlet kernels for large graph comparison". In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Ed. by D. van Dyk and M. Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 488–495.

[343] N. Shervashidze et al. "Weisfeiler-lehman graph kernels". In: *Journal of Machine Learning Research* 12.Sep (2011), pp. 2539–2561.

[344] Y. Shimizu et al. "Cell-free translation reconstituted with purified components". In: *Nature biotechnology* 19.8 (2001), p. 751.

[345] P. K. Singh. "Similar Vague Concepts Selection Using Their Euclidean Distance at Different Granulation". In: *Cognitive Computation* 10.2 (2018), pp. 228–241.

[346] D. L. Smith and Z. Zhang. "Probing noncovalent structural features of proteins by mass spectrometry". In: *Mass Spectrometry Reviews* 13.5-6 (1994), pp. 411–429. DOI: 10.1002/mas.1280130503.

[347] R. J. Smith et al. "Capture of endothelial cells under flow using immobilized vascular endothelial growth factor". In: *Biomaterials* 51 (2015), pp. 303 –312. ISSN: 0142-9612. DOI: 10.1016/j.biomaterials.2015.02.025.

[348] G. Solinas et al. "JNK1 in Hematopoietically Derived Cells Contributes to Diet-Induced Inflammation and Insulin Resistance without Affecting Obesity". In: *Cell Metabolism* 6.5 (2007), pp. 386 –397. ISSN: 1550-4131. DOI: 10.1016/j.cmet.2007.09.011.

[349] C. Song, S. Havlin, and H. A. Makse. "Origins of fractality in the growth of complex networks". In: *Nature Physics* 2.4 (2006), p. 275. DOI: 10.1038/nphys266.

[350] C. Song, S. Havlin, and H. A. Makse. "Self-similarity of complex networks". In: *Nature* 433.7024 (2005), p. 392. DOI: 10.1038/nature03248.

[351] S. Sonnenburg et al. "Large scale multiple kernel learning". In: *Journal of Machine Learning Research* 7.Jul (2006), pp. 1531–1565.

[352] A Suárez-Causado et al. "HGF/c-Met signaling promotes liver progenitor cell migration and invasion by an epithelial–mesenchymal transition-independent, phosphatidyl inositol-3 kinase-dependent pathway in an in vitro model". In: *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research* 1853.10 (2015), pp. 2453–2463. DOI: 10.1016/j.bbamcr.2015.05.017.

[353] G. J. Szekely and M. L. Rizzo. "Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method". In: *Journal of Classification* 22.2 (2005), pp. 151–183. ISSN: 1432-1343. DOI: 10.1007/s00357-005-0012-9.

[354] A. Tausz, M. Vejdemo-Johansson, and H. Adams. "JavaPlex: A research software package for persistent (co)homology". In: *Proceedings of ICMS 2014*. Ed. by H. Hong and C. Yap. Lecture Notes in Computer Science 8592. Software available at http://appliedtopology.github.io/javaplex/. 2014, pp. 129–136.

[355] The UniProt Consortium. "UniProt: the universal protein knowledgebase". In: *Nucleic Acids Research* 45.D1 (2017), pp. D158–D169. DOI: 10.1093/nar/gkw1099.

[356] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. 4th ed. Academic Press, 2008.

[357] R. L. Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (1953), pp. 267–276. ISSN: 1860-0980. DOI: 10.1007/BF02289263.

[358] D. R. Tocher. "Metabolism and Functions of Lipids and Fatty Acids in Teleost Fish". In: *Reviews in Fisheries Science* 11.2 (2003), pp. 107–184. DOI: 10.1080/713610925.

[359] E. Tomita, A. Tanaka, and H. Takahashi. "The worst-case time complexity for generating all maximal cliques and computational experiments". In: *Theoretical Computer Science* 363.1 (2006), pp. 28 –42. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2006.06.015.

[360] K. Tun et al. "Metabolic pathways variability and sequence/networks comparisons". In: *BMC bioinformatics* 7.1 (2006), p. 24. DOI: 10.1186/1471-2105-7-24.

[361] P. J. Turnbaugh et al. "A core gut microbiome in obese and lean twins". In: *Nature* 457.7228 (2009), p. 480. DOI: 10.1038/nature07540.

[362] J. M. Valtorta et al. "A Clustering Approach for Profiling LoRaWAN IoT Devices". In: *Ambient Intelligence*. Ed. by I. Chatzigiannakis, B. De Ruyter, and I. Mavrommati. Cham: Springer International Publishing, 2019, pp. 58–74. ISBN: 978-3-030-34255-5. DOI: 10.1007/978-3-030-34255-5_5.

[363] M. Vassura et al. "Reconstruction of 3D Structures From Protein Contact Maps". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 5.3 (2008), pp. 357–367. ISSN: 1545-5963. DOI: 10.1109/TCBB.2008.27.

[364] L. Vendramin, P. A. Jaskowiak, and R. J. G. B. Campello. "On the Combination of Relative Clustering Validity Criteria". In: *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. SSDBM. Baltimore, Maryland, USA: ACM, 2013, 4:1–4:12. ISBN: 978-1-4503-1921-8. DOI: 10.1145/2484838.2484844.

[365] M. Vijayabaskar and S. Vishveshwara. "Interaction Energy Based Protein Structure Networks". In: *Biophysical Journal* 99.11 (2010), pp. 3704 –3715. ISSN: 0006-3495. DOI: 10.1016/j.bpj.2010.08.079.

[366] S. V. N. Vishwanathan et al. "Graph kernels". In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1201–1242.

[367] S. A. Vlahopoulos et al. "Dynamic aberrant NF-κB spurs tumorigenesis: A new model encompassing the microenvironment". In: *Cytokine & Growth Factor Reviews* 26.4 (2015). SI: Cytokines and growth factors in cancer biology, pp. 389 –403. ISSN: 1359-6101. DOI: 10.1016/j.cytogfr.2015.06.001.

[368] F. Wang and J. Sun. "Survey on distance metric learning and dimensionality reduction in data mining". In: *Data Mining and Knowledge Discovery* 29.2 (2015), pp. 534–564. ISSN: 1573-756X. DOI: 10.1007/s10618-014-0356-z.

[369] F. Wang et al. "An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity". In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by P. Perner. Cham: Springer International Publishing, 2017, pp. 291–305. ISBN: 978-3-319-62416-7. DOI: 10.1007/978-3-319-62416-7_21.

[370] K. Wang, S. I. Grivennikov, and M. Karin. "Implications of anti-cytokine therapy in colorectal cancer and autoimmune diseases". In: *Annals of the Rheumatic Diseases* 72.suppl 2 (2013), pp. ii100–ii103. ISSN: 0003-4967. DOI: 10.1136/annrheumdis-2012-202201.

[371] J. H. Ward Jr. "Hierarchical Grouping to Optimize an Objective Function". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. DOI: 10.1080/01621459.1963.10500845.

[372] S. Warshall. "A Theorem on Boolean Matrices". In: *J. ACM* 9.1 (1962), pp. 11–12. ISSN: 0004-5411. DOI: 10.1145/321105.321107.

[373] S. P. Wasser. "Medicinal Mushroom Science: History, Current Status, Future Trends, and Unsolved Problems". In: *International Journal of Medicinal Mushrooms* 12.1 (2010), pp. 1–16. ISSN: 1521-9437. DOI: 10.1615/IntJMedMushr.v12.i1.10.

[374] L. Wasserman. "Topological Data Analysis". In: *Annual Review of Statistics and Its Application* 5.1 (2018), pp. 501–532. DOI: 10.1146/annurev-statistics-031017-100045.

[375] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. New York, USA: Cambridge University Press, 1994.

[376] D. J. Watts and S. H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), p. 440.

[377] W. Weaver. "Science and Complexity". In: *American Scientist* 36.4 (1948), p. 536.

[378] E. C. Webb. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* 6th ed. Academic Press, 1992.

[379] C. L. Webber Jr. et al. "Elucidating protein secondary structures using alpha-carbon recurrence quantifications". In: *Proteins: Structure, Function, and Bioinformatics* 44.3 (2001), pp. 292–303. DOI: 10.1002/prot.1094.

[380] R. A. Weinstein and B. Hota. "Contamination, disinfection, and cross-colonization: are hospital surfaces reservoirs for nosocomial infection?" In: *Clinical infectious diseases* 39.8 (2004), pp. 1182–1189. DOI: 10.1086/424667.

[381] M. Weis and F. Naumann. "Detecting Duplicates in Complex XML Data". In: *22nd International Conference on Data Engineering (ICDE'06)*. 2006, pp. 109–109. DOI: 10.1109/ICDE.2006.49.

[382] B. Wickstead and K. Gull. "The evolution of the cytoskeleton". In: *The Journal of Cell Biology* 194.4 (2011), pp. 513–525. ISSN: 0021-9525. DOI: 10.1083/jcb.201102065.

[383] P. G. Wolynes. "Evolution, energy landscapes and the paradoxes of protein folding". In: *Biochimie* 119 (2015), pp. 218 –230. ISSN: 0300-9084. DOI: 10.1016/j.biochi.2014.12.007.

[384] S. Wuchty. "Scale-Free Behavior in Protein Domain Networks". In: *Molecular Biology and Evolution* 18.9 (2001), pp. 1694–1702. DOI: 10.1093/oxfordjournals.molbev.a003957.

[385] K. Wüthrich. "Protein structure determination in solution by NMR spectroscopy". In: *Journal of Biological Chemistry* 265.36 (1990), pp. 22059–22062.

[386] K. Wüthrich. "The way to NMR structures of proteins". In: *Nature Structural & Molecular Biology* 8.11 (2001), p. 923.

[387] B. Xiao and E. R. Hancock. "Graph clustering using heat content invariants". In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer. 2005, pp. 123–130. DOI: 10.1007/11492542_16.

[388] B. Xiao, E. R. Hancock, and R. C. Wilson. "Graph characteristics from the heat kernel trace". In: *Pattern Recognition* 42.11 (2009), pp. 2589–2606. DOI: 10.1016/j.patcog.2008.12.029.

[389] X. Xu, J. Jäger, and H.-P. Kriegel. "A Fast Parallel Clustering Algorithm for Large Spatial Databases". In: *Data Mining and Knowledge Discovery* 3.3 (1999), pp. 263–290. ISSN: 1573-756X. DOI: 10.1023/A:1009884809343.

[390] W. Yan et al. "The construction of an amino acid network for understanding protein structure and function". In: *Amino Acids* 46.6 (2014), pp. 1419–1439. ISSN: 1438-2199. DOI: 10.1007/s00726-014-1710-6.

[391] P. Yanardag and S. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, Australia: ACM. New York, NY, USA, 2015, pp. 1365–1374. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2783417.

[392] Y. Yang et al. "Gene co-expression network analysis reveals common system-level properties of prognostic genes across cancer types". In: *Nature communications* 5 (2014), p. 3231. DOI: 10.1038/ncomms4231.

[393] Y. Yao. "A triarchic theory of granular computing". In: *Granular Computing* 1.2 (2016), pp. 145–157.

[394] Y. Yao. "The rise of granular computing". In: *Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)* 20.3 (2008), pp. 299–308.

[395] W. J. Youden. "Index for rating diagnostic tests". In: *Cancer* 3.1 (1950), pp. 32–35.

[396] L. Yujian and L. Bo. "A Normalized Levenshtein Distance Metric". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1091–1095. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1078.

[397] L. A. Zadeh. "Soft computing and fuzzy logic". In: *IEEE Software* 11.6 (1994), pp. 48–56. ISSN: 0740-7459. DOI: 10.1109/52.329401.

[398] L. A. Zadeh. "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic". In: *Fuzzy sets and systems* 90.2 (1997), pp. 111–127.

[399] M. Zaharia et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association. 2012, pp. 2–2.

[400] M. Zaharia et al. "Spark: Cluster Computing with Working Sets." In: *HotCloud* 10.10-10 (2010), p. 95.

[401] C. Zhan, G. Chen, and L. F. Yeung. "On the distributions of Laplacian eigenvalues versus node degrees in complex networks". In: *Physica A: Statistical Mechanics and its Applications* 389.8 (2010), pp. 1779 –1788. ISSN: 0378-4371. DOI: 10.1016/j.physa.2009.12.005.

[402] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. SIGMOD '96. Montreal, Quebec, Canada: ACM. New York, NY, USA, 1996, pp. 103–114. ISBN: 0-89791-794-4.

[403] W. Zhang, D. Zhao, and X. Wang. "Agglomerative clustering via maximum incremental path integral". In: *Pattern Recognition* 46.11 (2013), pp. 3056 – 3065. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2013.04.013.

[404] W. Zhang et al. "Graph Degree Linkage: Agglomerative Clustering on a Directed Graph". In: *Computer Vision – ECCV 2012*. Ed. by A. Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 428–441. ISBN: 978-3-642-33718-5. DOI: 10.1007/978-3-642-33718-5_31.

[405] X. Zhang, R. R. Nadakuditi, and M. E. J. Newman. "Spectra of random graphs with community structure and arbitrary degrees". In: *Phys. Rev. E* 89 (4 2014), p. 042816. DOI: 10.1103/PhysRevE.89.042816.

[406] Z. Zhang et al. "Exact eigenvalue spectrum of a class of fractal scale-free networks". In: *EPL (Europhysics Letters)* 99.1 (2012), p. 10007. DOI: 10.1209/0295-5075/99/10007.

[407] D. Zhao and X. Tang. "Cyclizing Clusters via Zeta Function of a Graph". In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller et al. Curran Associates, Inc., 2009, pp. 1953–1960.

[408] W. Zhao, H. Ma, and Q. He. "Parallel K-Means Clustering Based on MapReduce". In: *Cloud Computing*. Ed. by M. G. Jaatun, G. Zhao, and C. Rong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 674–679. ISBN: 978-3-642-10665-1. DOI: 10.1007/978-3-642-10665-1_71.

[409] J. Zhu et al. "1-norm support vector machines". In: *Advances in neural information processing systems*. 2004, pp. 49–56.

[410] A. Zomorodian. "Fast construction of the Vietoris-Rips complex". In: *Computers & Graphics* 34.3 (2010). Shape Modelling International (SMI) Conference 2010, pp. 263 –271. ISSN: 0097-8493. DOI: 10.1016/j.cag.2010.03.007.

[411] A. Zomorodian. "Topological data analysis". In: *Advances in applied and computational topology* 70 (2012), pp. 1–39.

[412] A. Zomorodian and G. Carlsson. "Computing Persistent Homology". In: *Discrete & Computational Geometry* 33.2 (2005), pp. 249–274. ISSN: 1432-0444. DOI: 10.1007/s00454-004-1146-y.