
Machine learning in complex networks: modeling, analysis, and applications

Thiago Christiano Silva

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Machine learning in complex networks: modeling, analysis, and applications

Thiago Christiano Silva

Advisor: Prof. Dr. Zhao Liang

Doctoral dissertation submitted to the *Instituto de Ciências Matemáticas e de Computação - ICMC-USP*,
in partial fulfillment of the requirements for the degree
of the Doctorate Program in Computer Science and
Computational Mathematics. *EXAMINATION BOARD
PRESENTATION COPY*.

USP – São Carlos
November 2012

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

S586m Silva, Thiago Christiano
Machine learning in complex networks: modeling,
analysis, and applications / Thiago Christiano
Silva; orientador Zhao Liang. -- São Carlos, 2012.
284 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2012.

1. Machine learning. 2. Complex networks. 3.
Competitive learning. 4. High level classification.
5. Random walks. I. Liang, Zhao, orient. II. Título.

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

Aprendizado de máquina em redes complexas: modelagem, análise e aplicações

Thiago Christiano Silva

Orientador: Prof. Dr. Zhao Liang

Tese apresentada ao Instituto de Ciências Matemáticas
e de Computação - ICMC-USP, como parte dos
requisitos para obtenção do título de Doutor em
Ciências - Ciências de Computação e Matemática
Computacional. *EXEMPLAR DE DEFESA.*

USP – São Carlos
Novembro de 2012

Acknowledgements

Firstly, I would like to thank my wife, Ester Cordeiro Silva, for her unwavering support and unconditional friendship in the good and hard times of my life, for her kindness and endless patience, and for her good spirit that has guided and still guides me through the deceiving pitfalls of life. More important than all, I especially thank her for gifting me with our precious daughter, Yasmin Christiane Silva. Both have provided me with countless moments of happiness, inspiration, and true love.

I cannot express my true feelings to my mother and my father for always being a guiding wheel and guardian in my life, for the wonderful provided moments, and for the eternal trust that enabled me to reach the position where I am today. I am truly honored for all that and forever grateful.

No less than that, I would like to express my sincere gratitude towards my grandfather and grandmother, who have always been at my side since I know myself. For what you have done in my life, it will be eternally remembered.

For my friends, I owe thanks for the good moments spent together, for the good talks that rendered several hours, and for the adventures that we have been through.

For FAPESP, I sincerely give my thanks for the continuous financial support, which opened doors to the development and execution of this project.

I would like to express my gratitude to the University of São Paulo and the Institute of Mathematics and Computer Sciences for hosting me through the B.E. degree and Doctorate period. I, here, say thanks to the professors that passed through this long adventure of mine, instructing and educating me. Without doubts, this, as a whole, has been one of the pillars that brought me where I am today.

I would of course like to express my deep gratitude to my thesis advisor, Zhao Liang, who has provided invaluable substantive and golden advice during the infinity quantity of good conversations that we have been through. These frequent discussions and the way that he has trusted me in our accomplishments have passionate me with research. I would say that our collective thinking has not only rendered good or bad ideas used in our works, but enabled me to capture and learn how to think in terms of scientific research. For me, this is a gorgeous achievement. In addition, I would like to emphasize that our friendship has solidified way beyond academical backgrounds and, for that, I will be eternally grateful for his good will in helping me in moments where I really needed most.

Lastly, I thank God for protecting me and pervading my life with good physical and mental health. Also, I praise Him for putting in my life the good people that I have met since then. I am truly appreciated for all these wonderful gifts.

Abstract

Machine learning is evidenced as a research area with the main purpose of developing computational methods that are capable of learning with their previously acquired experiences. Although a large amount of machine learning techniques has been proposed and successfully applied in real systems, there are still many challenging issues, which need be addressed. In the last years, an increasing interest in techniques based on complex networks (large-scale graphs with nontrivial connection patterns) has been verified. This emergence is explained by the inherent advantages provided by the complex network representation, which is able to capture the spatial, topological and functional relations of the data. In this work, we investigate the new features and possible advantages offered by complex networks in the machine learning domain. In fact, we do show that the network-based approach really brings interesting features for supervised, semisupervised, and unsupervised learning. Specifically, we reformulate a previously proposed particle competition technique for both unsupervised and semisupervised learning using a stochastic nonlinear dynamical system. Moreover, an analytical analysis is supplied, which enables one to predict the behavior of the proposed technique. In addition to that, data reliability issues are explored in semisupervised learning. Such matter has practical importance and is found to be of little investigation in the literature. With the goal of validating these techniques for solving real problems, simulations on broadly accepted databases are conducted. Still in this work, we propose a hybrid supervised classification technique that combines both low and high orders of learning. The low level term can be implemented by any classification technique, while the high level term is realized by the extraction of features of the underlying network constructed from the input data. Thus, the former classifies the test instances by their physical features, while the latter measures the compliance of the test instances with the pattern formation of the data. Our study shows that the proposed technique not only can realize classification according to the semantic meaning of the data, but also is able to improve the performance of traditional classification techniques. Finally, it is expected that this study will contribute, in a relevant manner, to the machine learning area.

Keywords: particle competition, competitive learning, unsupervised learning, semisupervised learning, supervised learning, data clustering, data classification, high level classification, random walks, complex networks.

Resumo

Aprendizado de máquina figura-se como uma área de pesquisa que visa a desenvolver métodos computacionais capazes de “aprender” com a experiência. Embora uma grande quantidade de técnicas de aprendizado de máquina foi proposta e aplicada, com sucesso, em sistemas reais, existem ainda inúmeros problemas desafiantes que necessitam ser explorados. Nos últimos anos, um crescente interesse em técnicas baseadas em redes complexas (grafos de larga escala com padrões de conexão não triviais) foi verificado. Essa emergência é explicada pelas inerentes vantagens que a representação em redes complexas traz, sendo capazes de capturar as relações espaciais, topológicas e funcionais dos dados. Nesta tese, serão investigadas as possíveis vantagens oferecidas por redes complexas quando utilizadas no domínio de aprendizado de máquina. De fato, será mostrado que a abordagem por redes realmente proporciona melhorias nos aprendizados supervisionado, semissupervisionado e não supervisionado. Especificamente, será reformulada uma técnica de competição de partículas para o aprendizado não supervisionado e semissupervisionado por meio da utilização de um sistema dinâmico estocástico não linear. Em complemento, uma análise analítica de tal modelo será desenvolvida, permitindo o entendimento evolutional do modelo no tempo. Além disso, a questão de confiabilidade de dados será investigada no aprendizado semissupervisionado. Tal tópico tem importância prática e é pouco estudado na literatura. Com o objetivo de validar essas técnicas em problemas reais, simulações computacionais em bases de dados consagradas pela literatura serão conduzidas. Ainda nesse trabalho, será proposta uma técnica híbrida de classificação supervisionada que combina tanto o aprendizado de baixo como de alto nível. O termo de baixo nível pode ser implementado por qualquer técnica de classificação tradicional, enquanto que o termo de alto nível é realizado pela extração das características de uma rede construída a partir dos dados de entrada. Nesse contexto, aquele classifica as instâncias de teste segundo qualidades físicas, enquanto que esse estima a conformidade da instância de teste com a formação de padrões dos dados. Os estudos aqui desenvolvidos mostram que o método proposto pode melhorar o desempenho de técnicas tradicionais de classificação, além de permitir uma classificação de acordo com o significado semântico dos dados. Enfim, acredita-se que este estudo possa gerar contribuições relevantes para a área de aprendizado de máquina.

Palavras-chave: competição de partículas, aprendizado competitivo, aprendizado não supervisionado, aprendizado semissupervisionado, aprendizado supervisionado, agrupamento de dados, classificação de dados, classificação em alto nível, caminhadas aleatórias, redes complexas.

Contents

Acknowledgements	vii
Abstract	ix
Resumo	xi
Contents	xiii
List of Figures	xvii
List of Symbols	xxi
1 Introduction	1
1.1 Objectives	4
1.2 Motivations	8
1.3 Organization of the Remainder of the Document	12
2 Fundamental Concepts	15
2.1 Complex Networks	15
2.1.1 Basic Concepts of Graphs	16
2.1.2 Evolution Line and Historical Trends	18
2.1.3 Complex Network Models	20
2.1.4 Community Detection	25
2.1.5 Complex Network Measures	29
2.2 Machine Learning	34
2.2.1 Overview of Machine Learning	35
2.2.2 Network-based Unsupervised Learning	36
2.2.3 Network-based Supervised Learning	44
2.2.4 Network-based Semisupervised Learning	52
2.3 Chapter Remarks	61
3 Unsupervised Stochastic Competitive Learning in Complex Networks	63
3.1 Model Description	64
3.1.1 A Brief Overview of the Model	64
3.1.2 Deriving the Competitive Transition Matrix of the Dynamical System	65
3.1.3 Defining the Stochastic Nonlinear Dynamical System	74

3.1.4	Setting up the Initial Conditions of the Dynamical System	75
3.1.5	Detailing the Algorithm	75
3.1.6	Discovering the Computational Complexity	77
3.1.7	Method for Determining the Optimal Number of Particles	80
3.1.8	Method for Detecting Overlapping Structures	83
3.2	Empirical Analysis of the Technique	84
3.2.1	Parameter Sensitivity Analysis	84
3.2.2	Convergence Analysis	88
3.2.3	Impact Analysis of Different Similarity Functions in the Network Formation Step	92
3.3	Theoretical Analysis of the Technique	95
3.3.1	Mathematical Analysis	95
3.3.2	The Proposed Competitive System Generalizes Multiple Independent Random Walks	107
3.3.3	A Numerical Example	109
3.4	Computer Simulations	113
3.4.1	Computational Complexity Analysis	114
3.4.2	Simulations on Artificial Data	114
3.4.3	Simulations for Community Detection on Artificial and Real-World Networks	117
3.4.4	Simulations for Data Clustering on Artificial and Real-World Data Sets	120
3.4.5	Simulations for Detecting Overlapping Vertices and Communities	126
3.5	Application: Handwritten Digits and Letters Clustering	132
3.5.1	Motivation	132
3.5.2	The Network Formation Technique	133
3.5.3	Brief Information of the Handwritten Digits and Letters Data Sets	134
3.5.4	Determining the Optimal Number of Particles and Clusters	135
3.5.5	Handwritten Data Clustering	135
3.6	Chapter Remarks	140
4	Semisupervised Stochastic Competitive Learning in Complex Networks	143
4.1	Model Description	144
4.1.1	A First Glimpse at the Differences of the Semisupervised and the Unsupervised Versions	144
4.1.2	Familiarizing with the Environment	145
4.1.3	Deriving the Competitive Transition Matrix of the Dynamical System	145
4.1.4	The Initial Conditions of the System	146
4.1.5	Discovering the Computational Complexity	147
4.2	Theoretical Results	148
4.2.1	Mathematical Analysis	148
4.2.2	A Numerical Example	152
4.2.3	Validation of the Theoretical Results	155
4.3	Computer Simulations	156
4.3.1	Parameter Sensitivity Analysis	156
4.3.2	Simulations on Synthetic Data Sets	159
4.3.3	Computer Simulations on Benchmarked Data Sets	163

4.4	Application: Detection and Prevention of Error Propagation via Competitive Learning	173
4.4.1	Motivation	175
4.4.2	Detecting Incorrectly Labeled Vertices	176
4.4.3	Preventing the Label Propagation from Incorrectly Labeled Vertices	179
4.4.4	The New Competitive Learning System	180
4.4.5	Understanding the Properties and Dynamics of the New Model .	181
4.4.6	Computer Simulations	190
4.5	Chapter Remarks	197
5	Supervised High Level Data Classification in Complex Networks	199
5.1	Motivation	200
5.2	Model Description	203
5.2.1	Glimpsing over the Fundamentals behind the Model	203
5.2.2	Deriving the Hybrid Classification Framework	207
5.3	Some Possible Ways of Composing the High Level Classifier	210
5.3.1	High Level Classifier using a Mixture of Complex Network Measures	211
5.3.2	High Level Classifier using Tourist Walks	213
5.4	Computer Simulations using the High Level Classifier as a Mixture of Complex Network Measures	218
5.4.1	Illustrative Examples	218
5.4.2	Simulations on Real-World Data Sets	230
5.4.3	Application: Handwritten Digits Recognition	234
5.5	Computer Simulations using the High Level Classifier based on Tourist Walks	236
5.5.1	Illustrative Examples	236
5.5.2	Parameter Sensitivity Analysis	240
5.5.3	Simulations on Real-World Data Sets	245
5.5.4	Application: Handwritten Digits Recognition	247
5.6	Chapter Remarks	249
6	Conclusions	253
6.1	Concluding Remarks	253
6.2	Publications during the Doctorate Period	257
6.3	Scientific Contributions	258
6.4	Future Works	261
	Bibliography	265

List of Figures

1.1	A network which presents four well-defined communities.	4
2.1	An example of random networks of Erdős and Rényi. (a) A network constructed by means of the random approach proposed by Erdős e Rényi; and (b) plot of the average degree distribution in a network consisting of $V = 6\,000$ and $p = 0.01$	21
2.2	Network behavior as we increase the parameter p , which is responsible for the relocation frequency of the edges.	23
2.3	Schematic of a scale-free network.	24
2.4	Schematic of a random clustered network.	26
2.5	Illustration of a tourist walk with $\mu = 1$. The red (dark gray) and green (light gray) dots represent visited and unvisited sites, respectively. The dashed lines indicate the transient part of the walk, whereas the continuous lines, the attractor of the walk.	34
2.6	Schematic of the three paradigms of the machine learning area. (a) unsupervised learning (a clustering task is presented in the figure); (b) semisupervised learning (a semisupervised classification task is displayed in the figure); and (c) supervised learning (a supervised classification task is depicted in the figure).	37
2.7	An example of data set where semisupervised learning techniques would lead to more robust results than supervised learning techniques. The dotted line denotes the decision boundary that would be probably output by a supervised learning method. The continuous line displays the same information for a semisupervised learning algorithm.	53
2.8	Motivation for the research on network-based semisupervised methods.	56
2.9	Process of coalescence of vertices s and r . After the merge, s consumes r and becomes a super-vertex. All neighbors of r are connected to s during this procedure.	61
3.1	A typical situation where the red (dark gray) particle, presently located at vertex 1, has to choose a neighbor to visit in the next iteration.	68
3.2	Illustration of the reanimation scheme.	70
3.3	Illustration of the reanimation procedure in a typical situation.	72
3.4	A flowchart of the methodology to iterate the stochastic dynamical system ϕ	76

3.5	Determination of the optimal number of particles K in random clustered networks	82
3.6	Determination of the optimal number of particles K (the actual number of clusters) in real-world data sets	82
3.7	Cluster detection accuracy vs. parameter λ	86
3.8	Cluster detection accuracy vs. parameter Δ	87
3.9	Scatter plot of artificial databases constituted by two groups	90
3.10	Convergence analysis of the particle competition algorithm when $\langle R(t) \rangle$ is used	90
3.11	Convergence analysis of the particle competition algorithm when $ N(t+1) - N(t) _\infty$ is used	92
3.12	An arbitrary network constructed with the purpose of obtaining the largest reachable or feasible entry of $N(t)$ for a given t	101
3.13	Comparison of the consumed time of the proposed technique and the Modularity algorithm	115
3.14	A simple network with 11 vertices distributed into 2 unbalanced communities	115
3.15	Behavior of $\bar{N}(t)$ as the system progresses in time (t)	116
3.16	Illustration of an artificial community detection process via particle competition	117
3.17	Another illustration of an artificial community detection process via particle competition	118
3.18	GN's community detection benchmark results obtained by the proposed technique	119
3.19	Comparison of three community detection algorithms using the benchmark of Lancichinetti <i>et. al.</i>	120
3.20	Community detection result of the Zachary's karate club network by using the proposed method	121
3.21	Data clustering of toy data sets with different cluster shapes	122
3.22	Result of the calculation of the overlapping index for all vertices in the Zachary's "karate club" network	128
3.23	Dolphin Social Network observed by Lusseau	129
3.24	Hugo's sprawling novel of crime and redemption in post restoration entitled <i>Les Misérables</i>	130
3.25	American College Football network	131
3.26	The Scientific Collaboration Network data set	132
3.27	Determination of the optimal number of particles K (the optimal number of clusters) in real-world data sets	136
3.28	A broad set of samples of that were classified as being member of the cluster representing the pattern "2"	139
3.29	A broad set of samples of that were classified as being member of the cluster representing the pattern "5"	139
3.30	A broad set of samples of that were classified as being member of the cluster representing the pattern "6"	139
3.31	A broad set of samples of that were classified as being member of the cluster representing the pattern "8"	139
4.1	A reanimation schematic of an exhausted particle	147

4.2	Comparison of the theoretical and empirical distributions.	157
4.3	Classification accuracy vs. λ	158
4.4	Classification accuracy vs. Δ	158
4.5	Illustration of an artificial classification process through competitive particle walking.	160
4.6	Evolutional behavior of the average class domination level imposed by the particles in the network.	161
4.7	A simple networked data set.	162
4.8	Evolutional behavior of the average domination level imposed by the 3 particles in the network.	162
4.9	Data Classification of toy data sets.	164
4.10	PCA projection of the benchmarked data sets.	169
4.11	Another networked data example.	177
4.12	Minimum number of times that a particle must get exhausted in order to (4.23) and (4.24) to hold.	179
4.13	Accuracy rate vs. α	183
4.14	Accuracy rate vs. τ	185
4.15	Synthetic networks with different strategies for labeling the training set.	188
4.16	Behavior of the accuracy rate of the model vs. the proportion of mislabeled instances (q), when random clustered networks with constant mixture are used.	190
4.17	First critical point vs. network mixture $z_{\text{out}}/\langle k \rangle$	192
4.18	Behavior of the classification error as the proportion of wrongly labeled vertices increases on two real-world data sets.	194
5.1	A simple example of a supervised data classification task where there exists a class with a clear pattern, in this case, the red ("circle") class.	200
5.2	Illustration of the graph formation technique consisting in a combination of the k -nearest neighbor and the ϵ -radius techniques.	205
5.3	Overview of the two phases of the supervised learning.	205
5.4	A high level data classification using a synthetic data set.	219
5.5	Another example of high level classification in a synthetic data set.	222
5.6	Behavior of the decision boundaries as ρ varies in the toy data of Fig. 5.5.	223
5.7	Behavior of the decision boundaries as ρ varies for a multiclass problem.	224
5.8	Analysis of the minimum value of the compliance term, ρ_{\min} , that classifies the triangle-shaped item as members of the red or "circle" class.	225
5.9	Impact analysis of the compliance term on the model's accuracy rate when Gaussian distributions are employed.	228
5.10	An analysis of the impact of the compliance term ρ on three different traditional low level techniques applied to the MNIST database.	235
5.11	A high level data classification task using tourist walks.	239
5.12	Minimum value of the compliance term using tourist walks, ρ_{\min} , that results in the correct classification of the test instances.	240
5.13	Analysis of accuracy rate vs. the compliance term with three distinct high level classifiers based on tourist walks.	242
5.14	Behavior of the transient and cycle lengths of the Iris data set.	244
5.15	Accuracy rate vs. μ_c for 3 different values of the compliance term ρ	245

5.16	A detailed analysis of the impact of the compliance term ρ on different traditional low level techniques applied to the MNIST database when the high level is based on tourist walks.	249
5.17	Illustration of the pattern formation impact in a subset of samples extracted from the MNIST data set.	250
5.18	Transient and cycle lengths and the corresponding variations due to the insertion of a test instance.	252

List of Symbols

Notation	Description
-----------------	--------------------

t	Index for time.
i, j	Indices for vertices in the network.
k	Index for a particle in the network.
u	Index for a network measure plugged into the high level classifier.
V	Number of vertices.
E	Number of edges.
a_{ij}	The edge weight from vertex i to j .
λ	Counterweights the amount of preferential and random walks performed by all particles.
ρ	Compliance term used for calibrating the decisions of the low and high level classifiers.
n	Number of data items (instances) in a learning process.
l	Number of labeled data items (instances) in a learning process.
u	Number of unlabeled data items (instances) in a learning process.
L	Number of possible classes (labels).
μ_c	Critical memory length of the tourist walks.
ω_{\min}	Minimum energy that a particle can have.
ω_{\max}	Maximum energy that a particle can have.
Δ	Fraction of gained/lost energy depending on which type of vertex a particle visits.

Notation	Description
m	Number of network measures plugged into the network-based high level classifier.
$\alpha(u)$	Weight for the u th network measure plugged into the network-based high level classifier.
$N_i^{(k)}(t)$	Number of visits that vertex i has received from particle k up to time t .
$p^{(k)}(t)$	Location of the k th particle in the network at time t .
$E^{(k)}(t)$	Particle k 's energy at time t .
$S^{(k)}(t)$	Indicator of the state of the k th particle: active or exhausted at time t .
$F_i^{(j)}$	Hybrid framework's decision for the i th test instance towards class j .
$L_i^{(j)}$	Low level classifier's decision for the i th test instance towards class j .
$H_i^{(j)}$	High level classifier's decision for the i th test instance towards class j .
$\Delta G_i^{(j)}(u)$	u th network measure's variation when i is inserted into class j .
$\Delta T_i^{(j)}(u)$	Transient length's variation when test instance i is inserted into class j .
$\Delta C_i^{(j)}(u)$	Cycle length's variation when test instance i is inserted into class j .
$p^{(j)}$	Size proportion of class j .
$\mathbb{P}_{\text{transition}}^{(k)}(t)$	Particle k 's transition matrix at time t .
$\mathbb{P}_{\text{rand}}^{(k)}$	Particle k 's random walk matrix (time-invariant).
$\mathbb{P}_{\text{pref}}^{(k)}(t)$	Particle k 's preferential walk matrix at time t .
$\mathbb{P}_{\text{rean}}^{(k)}(t)$	Particle k 's reanimation matrix at time t .
\mathcal{X}	Set of raw, vector-based data items.
$\mathcal{X}_{\text{training}}$	Training set of data items.
$\mathcal{X}_{\text{test}}$	Test set of data items.
\mathcal{V}	Set of vertices in the network.
\mathcal{V}_u	Set of unlabeled vertices in the network.
\mathcal{V}_l	Set of labeled vertices in the network.
\mathcal{Q}	Set of wrongly labeled vertices in the network.
\mathcal{E}	Set of edges in the network.
\mathcal{K}	Set of particles inserted into the network.
\mathcal{L}	Set of possible labels (classes).
\mathcal{I}_t	Set containing all irreducible fractions satisfying Lemma 3 at time t .
\mathcal{M}_t	Set of all $N(t)$ whose every entry is in \mathcal{I}_t .

CHAPTER

1

Introduction

Human beings are born with the fascinating gift of learning. With the aid of such ability, they absorb and assimilate knowledge throughout their entire life. In an attempt to simulate such notorious characteristic in a computational environment, the research area entitled *machine learning* arose, which aims at developing computational methods that are capable of “learning” with accumulated experiences (Bishop, 2007; Duda *et al.*, 2000; Mitchell, 1997). Based on the computational data representation obtained from a wide range of domains, the learning machine techniques can, in an automatic manner, generate models apt to organize the existing knowledge or, yet, mimic the behavior of a human expert. Generally speaking, these techniques are traditionally classified in two paradigms: supervised and unsupervised learning (Bishop, 2007; Duda *et al.*, 2000). In the *supervised learning*, the goal is to deduce concepts regarding the data from the labeled instances; i.e., the learning process tries to construct a mapping function conditioned to the provided training set. When the labels comprise discrete values, then the problem is denominated *classification*, whereas when the values are continuous, *regression*. On the other hand, in the *unsupervised learning*, the main task consists in grouping the data, based on some similarity criterion. The learning process, in this case, is guided by the provided data, since no prior knowledge about the existing classes is required (Mitchell, 1997).

Supervised learning requires a labeled data set for training. However, the task of manual labeling is, in the majority of the cases, a cumbersome and expensive process, which usually involves the work of human experts. In order to soften this shortcoming, the area denominated *semisupervised learning* was proposed, whose main distinct characteristic resides in the fact that both the labeled and the unlabeled data are used in the prediction process by means of, for example, a propagating process. Frequently

in real-world situations, the data sets consist of a great amount of unlabeled data and a few labeled data. In this way, semisupervised learning can considerably reduce human experts' efforts. Moreover, empirical results have shown that the usage of unlabeled data can improve the performance of the classifier (Chapelle *et al.*, 2006; Singh *et al.*, 2008).

Competition is a natural process observed in nature and in many social systems that have limited resources, such as water, food, mates, territory, recognition, etc. Competitive learning is an important machine learning approach which is widely employed in artificial neural networks to realize unsupervised learning. Early developments include the famous Self-Organizing Map (SOM - *Self-organizing Map*) (Kohonen, 1990), Differential Competitive Learning (Kosko, 1991), and Adaptive Resonance Theory (ART - *Adaptive Resonance Theory*) (Carpenter and Grossberg, 1987; Grossberg, 1987). From then on, many competitive learning neural networks have been proposed (Allinson *et al.*, 2001; Amorim *et al.*, 2007; Athinarayanan *et al.*, 2002; Jain *et al.*, 2010; Kaylani *et al.*, 2010; López-Rubio *et al.*, 2009; Lu and Ip, 2009; Meyer-Bäse and Thümmel, 2008; Príncipe and Miikkulainen, 2009; Tan *et al.*, 2008) and a wide range of applications has been considered. Some of these include data clustering, data visualization, pattern recognition, and image processing (Bacciu and Starita, 2008; Chen *et al.*, 2005; Deboeck and Kohonen, 2010; do Rêgo *et al.*, 2010; Liu *et al.*, 2008; Wang *et al.*, 2009; Xu and II, 2005). Without a doubt, competitive learning neural networks represent one of the main successes of the neural network theory development. However, at least two problems remain: (i) The constructed network is usually small, in such a way that competition occurs among a small number of neurons. Consequently, the model may not exhibit high robustness for data processing. (ii) There is not a direct connection between the input data and the trained competitive learning neural network. When a large data set is mapped into a network with a small number of neurons, it becomes hard to see the correspondence between the original data and the trained neural network. This is one of the reasons why neural networks sometimes are considered as "black box" systems.

A random walk is a mathematical formalization of a trajectory that consists of taking successive random steps (Pearson, 1905). It has been used to describe many natural phenomena and it has also been applied to solve a wide range of engineering problems. Some of these include graph matching and pattern recognition (Gori *et al.*, 2005), image segmentation (Grady, 2006), neural network modeling (Jiang and Wang, 2000; Liang *et al.*, 2009), network centrality measure (Noh and Rieger, 2004), network partition (Zhou, 2003a), construction and analysis of communication networks (Zeng *et al.*, 2010; Zhong *et al.*, 2008).

Over the last decade, there has been an increasing interest in network research, with the focus shifting away from the analysis of small graphs to the consideration of

large-scale graphs, called *complex networks*. Such networks have emerged as a unified representation of complex systems in various branches of science. In general, they are used to model systems which have a nontrivial topology and are composed of a large amount of vertices (Albert and Barabási, 2002; Barrat *et al.*, 2008; Newman, 2003b). Machine learning and data mining techniques using complex networks have triggered increased attention. This is because networks are ubiquitous in nature and everyday life, since many data sets are artlessly represented by them. Examples include the Internet, the World Wide Web, biological neural networks, social networks (between individuals and among companies and organizations), food webs, metabolic networks and distribution (such as the bloodstream, postal delivery routes and distribution of electric energy), etc. Many other kinds of data sets can be transformed into network representations. For instance, a set of items described by feature (or attribute) vectors can be transformed into a network by simply connecting each sample to its k nearest neighbors. In addition, the complex network representation unifies the structure, dynamics, and functions of a system which it represents. It does not only describe the interaction among vertices (structure) and the evolution of such interactions (dynamics), but also reveals how the structure and dynamics affect the overall function of the network (Newman, 2010). For example, it is well known that there is a strong connection between the structure of protein-protein interaction networks and protein functions (Palla *et al.*, 2005). The main motivation of graph theory research is the ability to describe the topological structure of the original system. In the machine learning domain, it has been shown that the topological structure is quite useful to detect clusters of arbitrary forms in data clustering (Fortunato, 2010; Karypis *et al.*, 1999).

The networks with complex topologies, in the mid-60, were traditionally described in accordance with the model proposed by (Erdős and Rényi, 1959), known as *Random Graphs*. In 1998, Watts and Strogatz discovered that, given a regular network, the mean value of its shortest paths could be drastically reduced by means of randomly modifying a small quantity of the network's links (Watts and Strogatz, 1998). The resulting network is referred to as *Small-World Network*. In 1999, Barabási and Albert concluded that many real-world networks share a vertex degree distribution following a power law: $P(k) \sim k^{-\gamma}$, in which k is the number of links emanating from an arbitrary vertex and γ is the scale exponent (Barabási and Albert, 1999). This heterogeneous distribution models the existence of a small group of vertices that present a high quantity of links. These networks are nominated *Scale-Free Networks*.

One of the striking phenomena of complex networks is the presence of *communities*. The notion of community in networks is straightforward: each community is defined as a subgraph whose vertices are densely connected within itself, but sparsely connected with the rest of the network. Figure 1.1 shows a network with the presence of four well-defined communities. Within the figure, it is clear to visualize that there

exists several connections among the vertices of the same community, while this quantity is scarce in the case of vertices of different communities. Community detection in complex networks has turned out to be an important topic in graph mining and data mining (Danon *et al.*, 2005; Fortunato, 2010; Newman and Girvan, 2004). In graph theory, community detection corresponds to graph partition, which has been shown to be an NP-complete problem (Fortunato, 2010). For this reason, a lot of efforts has been spent to develop efficient but suboptimal solutions, such as the spectral method (Newman, 2006a), the technique based on the “betweenness” measure (Newman and Girvan, 2004), modularity optimization (Newman, 2004), community detection based on the Potts model (Reichardt and Bornholdt, 2004), synchronization (Arenas *et al.*, 2006), information theory (Fortunato *et al.*, 2004), and random walks (Zhou, 2003a). For a recent review of this topic, see (Fortunato, 2010).

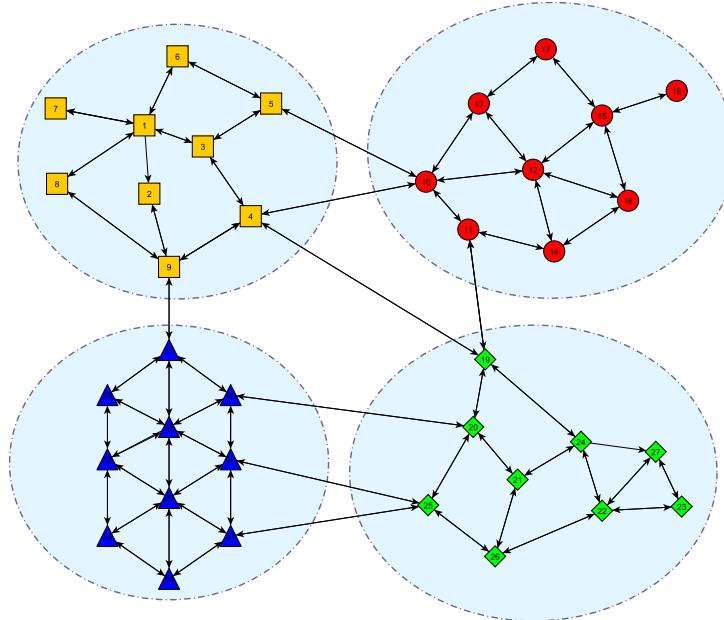


Figure 1.1: A network which presents four well-defined communities. The different vertices' colors or formats denote the communities to which each of them belong.

1.1 Objectives

This project brings as its main goal the development of new machine learning techniques based on complex networks. In this work, the three branches of the machine learning area are going to be tackled and investigated, i.e., the unsupervised, the semisupervised, and the supervised learning areas. Behind the aforementioned goal, the main concerns that this work tries to address are threefold:

- The development of new machine learning computational methods are expected, whenever possible, to be equipped with an underlying mathematical framework,

which is responsible for characterizing the behavior of these techniques in a concise and formal manner. We believe that this is an important step to better understand the dynamics of the models and, as a consequence, to better enable one to perceive the potentialities and the shortcomings offered by them. Nonetheless, whenever it is possible, empirical studies are likewise conducted to consolidate and to confirm the validity of our analytical predictions;

- The design of these machine learning techniques, in contrast to traditional methods, are expected to provide alternative and novel ways to solve the challenging problems posed by the machine learning area, such as novel and efficient algorithms modeled for the tasks of clustering and classification. In other words, we intend to investigate the new features and possible advantages offered by complex networks in the machine learning domain. In fact, we do show that the network-based approach really brings interesting features for machine learning;
- Having in mind the possibility of utilizing the methods described herein in real-world applications, we also take into account the design of techniques that are complementary in terms of accuracy rate and computational complexity. In practical terms, we must not build a technique that can provide excellent accuracy rates at the cost of high computational complexity. Conversely, a technique with low computational power is useless if its prediction power is not honorable. Therefore, we seek in-between solutions over these two extremes.

The specific goals and a brief summary of each topic comprising the current project are discussed as follows:

- With regard to the unsupervised learning area¹:
 - I. Competitive learning is an important machine learning approach which is widely employed in artificial neural networks. In view of this, we aim at proposing and developing an alternative unsupervised learning technique based on particle competition in complex networks. The model consists of several particles walking within the network and competing with each other to occupy as many vertices as possible, while attempting to reject intruder particles. The particle's walking rule is composed of a stochastic combination of random and preferential movements. The model has been applied to solve community detection and data clustering problems. As we will see in Chapter 3, computer simulations reveal that the proposed technique

¹Works related to this topic have been published in the IEEE Transactions on Neural Networks and Learning Systems ([Silva and Zhao, 2012a](#)), the Journal of Mathematical Imaging and Vision ([Silva et al., 2012c](#)), and submitted to the Information Sciences.

-
- presents high precision of community and cluster detections, as well as low computational complexity;
- II. The particle competition model is constructed under a stochastic nonlinear dynamical system. With this respect, we provide an analytical analysis of the model, deriving probabilistic expressions that are able to predict the model's behavior as time progresses. A numerical validation confirms the theoretical predictions. In addition, we show that the model generalizes the process of single random walks to multiple interacting random-preferential walks in a competitive way. Such generalization is realized by calibrating the parameters of the model;
 - III. A convergence analysis of the particle competition model is supplied. Therein, we show that the model does not converge to a fixed point, but instead it is confined within a certain region with a finite diameter. Furthermore, an upper bound of this region is estimated. In practical terms, we conclude that the particle competition model does not present asymptotic stability, but instead it presents structural stability. In our opinion, this characteristic is not a disadvantage, because it is more similar to several real-world systems on account of the noises and other uncontrolled variables presented by the real environment;
 - IV. In order to better enable the usage of the proposed model, a detailed parameter analysis is conducted;
 - V. An efficient method for estimating the most likely number of clusters is proposed. It uses an internal evaluator index that monitors the information generated by the competition process itself. Under some assumptions, we show that the cluster number determination process may not increase the model's complexity order. Since the determination of the actual number of clusters is an important issue in data clustering, our method also presents a contribution to this topic;
 - VI. A fuzzy index for detecting overlapping cluster or community structures in the network is proposed. As opposed to the majority of methods that calculate overlapping indices, our index calculation process is embedded within the competition process. As a consequence, we show that it also does not increase the model's complexity order.
 - VII. We apply the particle competition model in several real-world data sets in community detection and clustering tasks. Finally, we delve into an application of handwritten digits grouping and show that the model can supply reasonable results.

- With regard to the semisupervised learning area²:
 - I. We extend the particle competition model to the semisupervised learning area for classification tasks, as we will see in Chapter 4. In this case, generally, a small quantity of items is labeled. For each labeled item, a particle representing it is put into the network. A new factor in this technique is the possibility of cooperation among particles representing the same label. In practical terms, the particles of the same class cooperate among them, while the particles of different classes compete with each other to propagate class labels to the whole network;
 - II. Similarly to the unsupervised learning case, a mathematical analysis is also performed;
 - III. The reliability of the labels is a crucial factor in a semisupervised learning environment, because mislabeled samples may propagate wrong labels to a portion of or even the entire data set. Here, we address the error propagation problem originated by these mislabeled samples by presenting detection and prevention processes embedded within the competitive model. We show that this mechanism may be perfectly applied to autonomous systems, which can deeply suffer from these mislabeled vertices. The parameters involved in this detection and prevention process are also carefully analyzed.
- With regard to the supervised learning area³:
 - I. A hybrid framework composed of a convex combination of low and high level classifiers is proposed, as we will see in Chapter 5. Traditional supervised data classification considers only physical features (e.g., distance or similarity) of the input data. Here, this type of learning is called *low level classification*. On the other hand, the human (animal) brain performs both low and high orders of learning and it has facility in identifying patterns according to the semantic meaning of the input data. Data classification that considers not only physical attributes but also the pattern formation is here referred to as *high level classification*. The idea behind introducing this resides in the fact that data items often have patterns or organizational features that are left hidden within the numerous relationships among them. These, in turn, are not very well explored by traditional low level classifiers. The high

²Works related to this topic have been published in the IEEE Transactions on Neural Networks and Learning Systems ([Silva and Zhao, 2012b](#)) and accepted for publication in the Neural Networks ([Silva and Zhao, 2012f](#)).

³Works related to this topic have been published in the IEEE Transactions on Neural Networks and Learning Systems ([Silva and Zhao, 2012d](#)) and submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence.

level term precisely comes into play to fill in this gap. As we will see, in general terms, both classifiers are necessary to provide good results. Therefore, one alone may not yield the best results, but the synergy between both may boost the accuracy rate in some cases.

- II. Motivated by the intrinsic ability to describe topological structures among the data items, we propose two types of high level classifiers, all of which running in a networked environment. As a common goal, both realize the prediction scheme by extracting the features of the underlying network constructed from the input data. The high level classifiers are comprised of:
 - (a) We use the well-known assortativity, clustering coefficient, and average degree. We show that the combination of these three measures can capture from local to global characteristics of the network, when utilized with a good low level classifier.
 - (b) We apply a weighted combination of tourist walks with different memory lengths. For this end, we use the variations of the transient and cycle lengths of tourist walks for each value of the tourist's memory. We show that this is able to capture complex patterns of the network and is able to provide reasonable accuracy rates when utilized in conjunction with a good low level classifier.
- III. An interesting phenomenon uncovered in this work is that, as the class configuration's complexity increases, such as the mixture among different classes, a larger portion of the high level term is required to get correct classification. This feature confirms that the high level classification has a special importance in complex situations of classification.
- IV. The hybrid framework is employed in a real-world application, where it is capable of identifying variations and distortions of handwritten digits images. As a result, it supplies an improvement in the overall pattern recognition rate.

1.2 Motivations

The main motivation that surrounds this project comes from the fact that complex networks are powerful tools for several disciplines of science, including modeling and application of machine learning in data analysis. Due to the high complexity and versatility of this approach, there is still a large space for exploration. By virtue of this, our primary objective here is to incorporate these powerful tools into tasks of the unsupervised (data clustering or community detection), the semisupervised (semisupervised

classification), and the supervised learning (supervised classification and regression) areas, all of which pertaining to the machine learning area, with the goal of obtaining more robust systems. Sustained by the main goal, new machine learning techniques will be constructed under some motivations and inspired by the gaps encountered in the literature. In the next, we succinctly describe them.

One strong argument for delving into the process of particle competition is that it is very similar to many social and natural processes, such as: competition among animals, territorial exploration by humans (animals), election campaigns, among others. Furthermore, the random-preferential movement incorporated into the particles' movement policy can substantially improve the classification rate, as will be seen in the Chapters 3 and 4. This model corroborates the importance of the randomness role in evolutionary systems whose primary function is to prevent, in an automatic manner, the particles from falling into local traps. Besides that, it supplies, for the particles, the ability to explore unknown territories. Therefore, a certain amount of randomness is essential for the learning process. This randomness is charged with representing the state "I do not know" and lends itself as an effective "explorer of new features."

The particle competition model was originally proposed by ([Quiles et al., 2008](#)), where only a particle competition procedure was introduced without any formal definition. This technique displays at least two salient advantages in comparison to many other recent community detection techniques ([Boccaletti et al., 2007](#); [Danon et al., 2007](#); [Newman and Girvan, 2004](#); [Reichardt and Bornholdt, 2004](#); [Zhou, 2003b](#)):

- The mechanism of particle competition not only provides a community detection technique, but also presents a general scheme for competitive machine learning;
- The technique provides high community detection accuracy and, at the same time, low computational complexity.

In this project, a rigorous definition is provided, in which the particle competition is formally molded from a stochastic nonlinear dynamical system. Moreover, we not only will provide definitions for the particle competition in the unsupervised learning area, but also will extend and develop new mechanisms to adapt the same model for semisupervised learning tasks. One of these new mechanisms is the introduction of cooperation among particles. In short terms, particles of the same class proceed in the network in a cooperative manner to propagate their labels, while particles of different classes compete with each other to determine the class borders. Given that the model of several interacting particles corresponds to many natural and artificial systems, the study of this topic stands as an important task. Moreover, little investigation has been done in the literature for such kinds of systems and, in the few available works, several assumptions are taken to conduct the mathematical formalities ([Chau and Basu, 2011](#);

[Cooper et al., 2009](#)). For example, [Cooper et al. \(2009\)](#) analyze the problem of multiple random walks in regular graphs. [Chau and Basu \(2011\)](#) also explores these kinds of systems, but with a focus on intermittently connected networks. Therefore, due to relative scarcity of theory for such models, this work (the model definition *per se*) is an important step to understand and to dominate these systems. Another interesting feature is that the model has a local label-spreading fashion, i.e., due to the competitive mechanism, each particle only visits a portion of vertices potentially belonging to the current particle or its teammates. This can be roughly understood as a “divide-and-conquer” effect embedded in the competitive-cooperative scheme. In this way, many long-range redundant operations are avoided. As a result, the proposed method has a lower computational complexity order. Since the underlying network is constructed directly from the input data set, the correspondence between the input data and the processing result (the final network) is maintained. Consequently, the “black box” effect can be avoided at a large extent.

Traditional techniques of graph-based semisupervised learning involve minimizing a cost function and the number of matrix multiplications. Thus, the computational complexity of these techniques is usually of the order $\mathcal{O}(V^3)$ or higher ([Belkin and Niyogi, 2003; Belkin et al., 2004; Zhou et al., 2004](#)), where V is the number of vertices. Even though methods for enhancing matrix multiplication have been extensively studied (c.f. the generalized iterated matrix-vector multiplication technique, GMIV-M, proposed by ([Kang et al., 2011](#))), the minimization of the cost function, which is usually based on a regularization framework, slows down the whole process ([Zhou et al., 2004](#)). It is expected that the models generated based on competition of particles will be more efficient, which is important to treat large-scale databases. Therefore, the particle competition method in the semisupervised environment is going to be an endower to this direction, i.e., trying to fill in this gap in terms of the computational complexity.

Most of the traditional community detection methods aim at assigning each vertex to a single community ([Fortunato, 2010](#)). However, in real networks, vertices are often shared among communities ([Fortunato, 2010](#)). For example, in the language network composed of words as vertices, the word “Bright” might be a vertex member of several communities, such as communities representing words related to the following subjects: “Light,” “Astronomy,” “Color,” “Intelligence,” and so on ([Palla et al., 2005](#)). In a social network, each person naturally belongs to the communities of the company where he/she works and also to the community of his/her family at the same time. Therefore, uncovering overlapping community structure is important not only for network mining, but also for data analysis in general, once a data set is transformed into a network ([Evans and Lambiotte, 2009; Lancichinetti et al., 2009; Nicosia et al., 2009; Palla et al., 2005; Shen et al., 2009; Sun et al., 2011; Zhang et al., 2007](#)). A salient drawback of the majority of these techniques resides in the fact that the detection of the overlap-

ping characteristics of the input network is performed as a separated or dedicated process apart from the standard community detection technique. In this way, additional computational time is required. As a result, the whole process may have high computational complexity. As we will see, the particle competition technique detects overlapping cluster structures using the competition process itself, i.e, the detection procedure is already embedded into the model. As a result, we will verify that our method does not increase the model's complexity order. Considering that the determination of overlapping vertices is an important issue in community detection ([Palla et al., 2005](#)), our method also presents a contribution to this topic.

The quality of the training data is a fundamental issue in semisupervised learning because, in this case, less labeled data is available and errors (wrong labels) may easily be propagated to a portion of or the entire data set. Though this is an important topic, it has not received much attention from researchers and there are still few works devoted to the study of semisupervised learning from imperfect data ([Amini and Gallinari, 2003, 2005; Hartono and Hashimoto, 2007](#)). In this work, we will also treat the problem of error propagation in semisupervised learning using the particle competition method. Usually, in supervised or semisupervised learning, the input label information of the training data set is supposed to be completely reliable. However, in real situations, this is not always true and mislabeled samples are commonly found in the data sets due to instrumental errors, corruption from noise, or even human mistakes in the labeling process. For example, in a medical diagnostic system, the diagnostic results in the training set provided by doctors may be wrong. If these kinds of wrong labels are used to further classify new data (in the supervised learning case) or are propagated to the unlabeled data (in the semisupervised learning case), some severe consequences may occur. This situation becomes more critical in autonomous learning, where no external or minimal external intervention is involved. Thus, if the prior knowledge presented to the autonomous learning system contains errors, the performance of the learning system will get worse and worse because of the error propagation. Therefore, considering and designing mechanisms to prevent error propagation is important in the machine learning study and especially in the autonomous learning. Specifically, the prevention of error propagation can benefit the learning systems from two orthogonal aspects:

- i. Improvements of the performance of the learning system, i.e., the system can learn from errors;
- ii. Avoidance of a system's catastrophe by limiting the spreading of wrong labels (input and generated errors).

In this work, we present a novel mechanism to prevent error propagation in general semisupervised learning using the particle competition model. To our knowl-

edge, many semisupervised learning techniques have been proposed (Chapelle *et al.*, 2006), but the great majority considers that the label information of the labeled subset is totally correct, i.e., there is no error prevention mechanism. In this way, the proposed mechanism together with the analysis and numerical simulations presented in this work also make a clear contribution to general machine learning and especially to autonomous learning research.

In the field of supervised learning, techniques that predict using physical features but not the class pattern formation are called *low level classification* techniques. Usually, the data items are not isolated points in the attribute space, but instead tend to form certain patterns. The human (animal) brain performs both low and high orders of learning and it has facility in identifying patterns according to the semantic meaning of the input data. However, this kind of task, in general, is still cumbersome to be assessed by computers. Supervised data classification which not only considers physical attributes but also pattern formation is here referred to as *high level classification*. Motivated by the fact that the topological properties of the networks have been shown to be quite useful in data analysis and have not received much attention by the community, in this work, we propose a hybrid classifier as an endower to fill in this gap. Specifically, we propose a hybrid classification technique which combines the low and high levels of learning. Generally speaking, the low level classification can be implemented by any traditional classification technique. On the other hand, the high level classification exploits the complex topological properties of the underlying network constructed from the input data. As we will see, improvements of the low level classification can be achieved by the combination of the two levels of learning. The contributions of our work in the supervised learning area are twofold:

- i. Proposal of a customizable hybrid framework that encompasses a low and a high level classifier;
- ii. Definition of two network-based high level classifiers.

Even though the proposed high level classifiers are simple, they are able to enhance the accuracy rate of low level classifiers in a significant manner. Nonetheless, this serves as a motivation for further investigation of this branch of the machine learning area.

1.3 Organization of the Remainder of the Document

The remainder of this thesis is organized as follows. In Chapter 2, we elucidate the fundamental concepts that make way for the understanding of the developments derived from this work. With this respect, we review the basic definitions of the complex network and machine learning area. Once the elementary theory is presented,

we present the results derived from this work in Chapters 3, 4, and 5. Specifically, in Chapter 3, we explore the unsupervised learning domain by proposing a novel particle competition model. Several experiments and mathematical investigations are performed, as well as a real-world application (handwritten digits and letters clustering). In Chapter 4, the particle competition model is extended to the semisupervised learning domain. Likewise the previous chapter, we also display many experiments and mathematical investigations, as well as another real-world application (detection and prevention of mislabeled vertices). In Chapter 5, we delve into the supervised learning domain by proposing a hybrid framework that derives its decision based on a convex combination of low and high level classifiers. The proposed model is analyzed in an empirical way and significant results are obtained. A real-world application (handwritten digits recognition) is supplied. Enclosing this thesis, in Chapter 6, we draw the main conclusions of this work. Moreover, the scientific contributions derived from this project are listed, as well as possible future works. Lastly, we show the main papers that have been published during the Doctorate period.



CHAPTER
2

Fundamental Concepts

In this chapter, we present the fundamental concepts of *complex networks* and *machine learning*, which are related to the development of this thesis. For each of these branches, we supply examples of state-of-art methods, focusing on the potentialities and the shortcomings of each one of them. This enables one to better select the adequate method over the myriad of available techniques for the problem at his/her hands. In addition, we motivate the application of the complex networks theory into the machine learning domain, by elencating the principal advantages that a network representation of the data offers over a raw, vector-based representation, and by displaying how this data representation improves the overall performance of machine learning techniques. Since in here we deal with machine learning techniques that work in a networked environment, we restrict ourselves, when explaining the latter, to techniques that rely on networks to produce results (network-based techniques). Of course, in this case, we first supply an overall glimpse of the wide machine learning area and, afterwards, we adequately situate where the techniques that we are interested in reside.

2.1 Complex Networks

We start out by presenting the main concepts of networks. Afterwards, we delve into the evolution line and findings of complex networks. Following that, we discuss the main types and measures proposed in the complex networks literature. Then, we present the topic of community detection, which is a closely related task to data clustering. Finally, we review some well-known network measures.

2.1.1 Basic Concepts of Graphs

In this section, we present the main terminology of the graphs or networks theory. First, in this thesis, the words graphs and networks are used interchangeably. In the following, we present the formal definition of a graph (Bollobas, 1998; Diestel, 2006; Gross and Yellen, 1999).

Definition 1. Graph: A graph \mathcal{G} is defined as a pair $\langle \mathcal{V}, \mathcal{E} \rangle$, where \mathcal{V} is a finite nonempty set of vertices and \mathcal{E} is the set of edges between the vertices $\mathcal{E} \subseteq \{(u, v) \mid u, v \in \mathcal{V}\}$. Some special graphs are defined as follows:

- **Undirected Graph:** When the relation \mathcal{E} is symmetric, meaning that $\forall (u, v) \in \mathcal{E} \Rightarrow (v, u) \in \mathcal{E}$, it is said that the graph is undirected. In other terms, when there is an edge linking vertices u to v , so there will be a link from v to u .
- **Directed Graph (Digraph):** When the relation \mathcal{E} satisfies the following restriction: $\exists (u, v) \in \mathcal{E} \mid (v, u) \notin \mathcal{E}$, it is said that the graph is directed (digraph). In other terms, these kinds of graphs must have at least an arbitrary edge linking u to v , with an absence of the oppose link.
- **Graph with no self-loops:** When the relation \mathcal{E} is irreflexive, meaning that $\forall v \in \mathcal{V}, (v, v) \notin \mathcal{E}$, the graph is said to be free of self-loops. This means that there is no way of traveling to the same vertex in a single transition.
- **Graph with self-loops:** When the relation \mathcal{E} satisfies the following restriction $\exists v \in \mathcal{V}, (v, v) \in \mathcal{E}$, the graph is said to have self-loops. This means that one can travel back to the same vertex through an edge without leaving it.

There is a special type of graph known as weighted graph, whose definition is given as follows. Note that, for this type of graph, the same graph categories discussed in Definition 1 can be applied to it (Chung, 1997; Godsil and Royle, 2001).

Definition 2. Weighted Graph: A weighted graph \mathcal{G} is defined as a triple $\langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$, where \mathcal{V} and \mathcal{E} are the sets of vertices and edges, respectively, and \mathcal{W} is a matrix composing the edge weights. For example, the entry $W(u, v) = k, (u, v) \in \mathcal{E}$, fixes as k the weight of the edge linking vertices u to v . If $(u, v) \notin \mathcal{E} \Rightarrow W(u, v) = 0$.

Remark 1. When \mathcal{W} is a binary matrix, then the weighted graph reduces to an unweighted graph, which is the special graph supplied in Definition 1.

In the next, some common terms are introduced, which will be used throughout this thesis (Bollobas, 1998; Chung, 1997; Diestel, 2006; Godsil and Royle, 2001; Gross and Yellen, 1999).

Definition 3. Adjacent Vertices: Two vertices are called adjacent if they share a common edge, in which case the common edge is said to join the two vertices. An edge and a vertex on that edge are called incident. In digraphs, it may occur that vertex v is adjacent to u , but the opposite is not true, in the case that only a directed edge from u to v is present.

Definition 4. Neighborhood of a Vertex: The neighborhood of a vertex $v \in \mathcal{V}$ in a graph \mathcal{G} is the set of vertices adjacent to v . The neighborhood is denoted by $N(v)$. Note that the neighborhood does not include v itself.

Definition 5. Degree (Valency) of a Vertex: The degree of a vertex v is the total number of vertices adjacent to v . The degree of a vertex v is denoted by $k(v)$. We can equivalently define the degree of a vertex as the cardinality of its neighborhood and say that, for any vertex v , $k(v) = |N(v)|$.

Remark 2. In a directed graph, it is important to distinguish between in-degree and out-degree. Recall that any directed edge has two distinct ends: an origin (beginning) and a destination (end). Each destination is counted separately. In relation to a specific vertex, the sum of origin endpoints counts toward the out-degree and the sum of destination endpoints counts toward the in-degree.

Definition 6. Graph Path: We write a path P as an ordered list of edges: $P = \{(v_1, v_2), (v_2, v_3), \dots, (v_k, v_{k+1})\}$, such that no vertex and no edge is repeated. The first vertex of the first edge of a path is the path's origin and the second vertex of the last edge is the path's destination. Both path's origin and destination are called endpoints of the path. The length of a path is given by the cardinality of the sequence P , i.e., $|P|$. Moreover, P is a path in \mathcal{G} only if every entry of P is in \mathcal{E} .

Definition 7. Circuit: A circuit is a path which ends at the vertex it begins. Therefore, a self-loop is a circuit of length one.

Definition 8. Cycle: A cycle is a circuit in which no vertex, except the first, appears more than once.

Definition 9. Distance between vertices: The distance $d(u, v)$ between two vertices $u \in \mathcal{V}$ and $v \in \mathcal{V}$ is the length of the shortest path from u to v , considering all possible paths in \mathcal{G} from u to v . The distance between any vertex and itself is 0. If there is no path from u to v , then $d(u, v) = \infty$.

Definition 10. Diameter: The diameter of \mathcal{G} is the length of the largest distance in \mathcal{G} .

Definition 11. Vertex Eccentricity: The eccentricity of $v_1 \in \mathcal{V}$ is the largest distance from v_1 to any other vertex $v \in \mathcal{V} \setminus \{v_1\}$.

Definition 12. Clique: A clique in an undirected graph is a subset of vertices such that every two vertices in the subset are connected by an edge.

Definition 13. Graph Component: A graph component is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the remainder of the graph.

Definition 14. Connected Graph: A graph is connected if there is a path connecting every pair of vertices. Therefore, a connected graph has always a single component.

Definition 15. Regular Graph: A graph is regular if all the vertices of \mathcal{G} have the same degree. In particular, if the degree of each vertex is k , \mathcal{G} is said to be k -regular.

Definition 16. Complete Graph: A complete graph is a graph in which every two distinct vertices are joined by exactly one edge. The complete graph with n vertices is denoted by K_n .

Definition 17. Null Graph: A null graph is a graph containing no edges. The null graph with n vertices is denoted by N_n .

Definition 18. Bipartite Graph: A bipartite graph is a graph whose set of vertices \mathcal{V} can be split into two subsets, in such a way that each edge of the graph joins a vertex in the first set to a vertex in the second set.

Definition 19. Tree Graph: A tree is a connected graph which has no cycles.

Definition 20. Spanning Tree: If \mathcal{G} is a connected graph, the spanning tree in \mathcal{G} is a subgraph of \mathcal{G} which includes every vertex of \mathcal{G} and is also a tree graph.

2.1.2 Evolution Line and Historical Trends

The study of networks began with the development of the graph theory, inaugurated by Leonhard Euler in 1736 with the solution of the seven bridges problem of Königsberg, today Kaliningrad, Russia. The problem, much discussed at the time, recorded that there were seven bridges crossing the river Pregel, with two intermediate islands. The residents desired to know whether it was possible to cross all these seven bridges, without repetition, and return to the starting point. Euler demonstrated, in an analytical manner, for the Russian Academy of Sciences in St. Petersburg, that it was not possible to complete such a walk. For this end, he made use of a graphical representation consisting of points and curves connecting these points. It was the beginning of the formal representation of a graph or network, known until today, with vertices and edges. Thereafter, several researchers began studying this branch of research in search of new theories and applications (Newman, 2003b).

In fact, the first major step in the study of complex networks was driven by Paul Erdős and Alfréd Rényi, who analyzed a certain type of network, called *random networks*, in their work published in 1959. This investigation opened doors to a novel area of study - the theory of random networks -, which represents a mixture of the graph

and probabilistic theories to generate and analyze large-scale graphs ([Erdős and Rényi, 1959](#)).

Following the chronology, in 1967, Stanley Milgram decided to accept the challenge posed by Frigyes Karinthy, which, inspired by the conjectures of Guglielmo Marconi at 1909, dared one to find another person which could not be transitively connected by using at most five intermediate people ([Milgram, 1967](#)). And it was exactly because of this problem that the concept denominated *separation in six degrees* was born, which was the first seed for the study of the *small-world networks*. To address this challenge, Milgram conducted experiments in order to try to discover the probability of any two arbitrary people to know each other. For this, letters were sent to random people living in predetermined regions of the United States, whose inner content dealt with information about any other arbitrary person. If the person referred to in the letter was known by the reader, then he/she would mail the letter for the recipient. On the other hand, if he/she did not know, then the letter should be sent to someone else known. At the end of experiment, Milgram found that the average number of referrals of one person to another reached 5.5 people. He was, therefore, empirically discovering the property of small world, which states that even though there are millions of vertices interconnected in a social network, the average distance between them is only a small amount, in the example, 5.5 people ([Milgram, 1967](#)).

Despite of the findings of Milgram, it was only in the late 90s that surveys on this field were retaken. In 1998, Watts and Strogatz found that the average shortest paths in a network can be drastically reduced by a random alteration of few links, starting from a regular network ([Watts and Strogatz, 1998](#)). The resulting network is called small-world networks, which, as we have seen, had already been empirically discovered by Milgram. In 1999, Barabási and Albert discovered that many real networks have a degree distribution of vertices that obeys a power law: $P(k) \sim k^{-\gamma}$, where k is the number of connections of a randomly chosen vertex and γ is a scaling exponent ([Barabási and Albert, 1999](#)). This heterogeneous distribution describes the existence of a small number of vertices that has a large number of connections. Such networks are called *scale-free networks*.

Driven by the technological advances and also by the increasing number of data to be jointly analyzed, the complex networks area has emerged as a unifying topic in complex systems and is present in various branches of science ([Bornholdt and Schuster, 2003](#)). Structurally, complex networks are represented by a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where \mathcal{V} represents the set of vertices and \mathcal{E} , the set of edges. According to [Albert et al. \(2004\)](#), complex networks are models for systems in general, by virtue of having a non-trivial topology, besides being composed of a large number of vertices. Among some examples, which the network representation is perfectly plausible, these include: the Internet ([Faloutsos et al., 1999](#)), the World Wide Web ([Albert et al., 1999](#)), biological neural

networks (Sporns, 2002), social networks among individuals (Scott, 2000) and between companies and organizations (Mizruchi, 1982), food webs (Montoya and Solée, 2002), metabolic networks (Jeong *et al.*, 2000) and distribution as the bloodstream (West *et al.*, 1999), postal delivery routes and distribution of electricity (Albert *et al.*, 2004), etc. In accordance with Strogatz (2001), some inherent characteristics in this type of network are:

- *The structural complexity* - which translates into the difficulty of network visualization;
- *The evolution* - which marks the constant changes in the network structure due to the inclusion and removal of vertices and connections (Dorogovtsev and Mendes, 2003);
- *The diversity of connections* - because these connections between vertices can represent many variations, such as capacity, length, width and direction; and
- *The dynamical nature* - which affects, at a large scale, the states of a network, as can be construed as traffic information (Zhao *et al.*, 2007), occurrences of failures in communications (Zhao *et al.*, 2004, 2005, 2007), similarity relations between vertices, the distribution of functions (Newman, 2003b), among others.

2.1.3 Complex Network Models

With the expectation of studying topological properties that are linked to real networks, several network models have been proposed. Some of these models even have inspired an extensive study due to its features of great interest. As examples of important categories of networks, one can list: random networks, small-world networks, clustered random networks and scale-free networks. In the next sections, we review these models in detail.

Random Networks

In the article dated back to 1959, Erdős and Rényi (1959) showed a model which is able to generate random networks consisting of V vertices and E edges. Starting from V vertices completely disconnected (no edges in the network), the network is built from the gradual addition of L edges randomly created, in a such a way that self-looping is avoided. Another similar model sets V vertices in a network, and there is a probability $p > 0$ of connecting each possible pair of vertices. The latter model is widely recognized as the model of Erdős and Rényi. Figure 2.1a depicts an example of this type of network. Note that no spacial relation between the vertices is used. In this

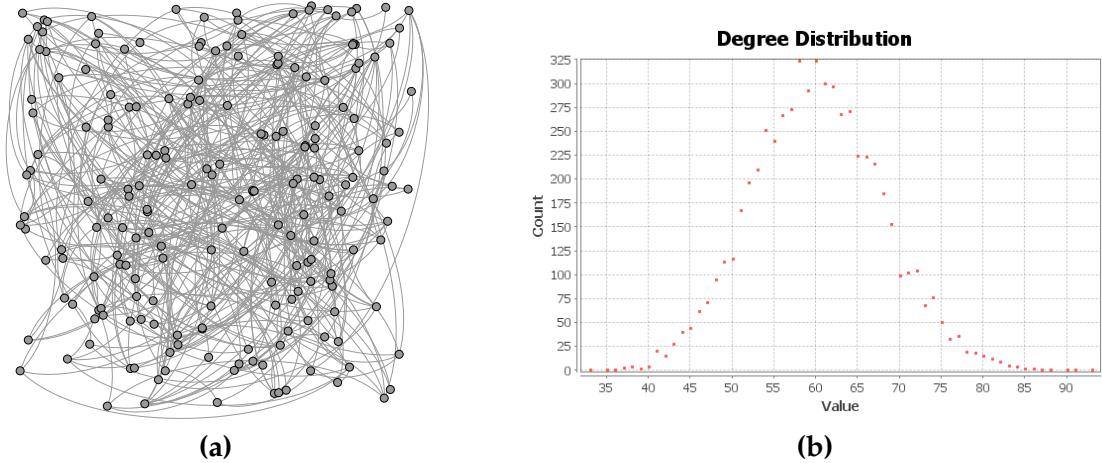


Figure 2.1: An example of random networks of Erdős and Rényi. (a) A network constructed by means of the random approach proposed by Erdős e Rényi; and (b) plot of the average degree distribution in a network consisting of $V = 6\,000$ and $p = 0.01$.

network formation, we merely create edges in a uniform probabilistic way, regardless of the similarity between vertices.

Since, for each vertex $i \in \mathcal{V}$ of the network (a total of V), there are $V - 1$ different possibilities of connections with other vertices, it follows that the cardinality of the sample space, $|\Omega|$, which quantifies the maximum number of feasible edges between the vertices, is given by:

$$|\Omega| = \frac{N(N - 1)}{2}, \quad (2.1)$$

where the division by 2 comes from the fact that we are considering that the graph is undirected, i.e., the edges are always bidirectional in relation to both linked vertices. In general, the presence of these two edges represents the occurrence of the same probabilistic event, on account of the inherent coupling (bidirectionally). Having in mind that an arbitrary edge will be present in a random network with probability p and will be absent with probability $1 - p$, and remembering that there are $\binom{V-1}{k}$ ways of choosing k vertices over $V - 1$ in total, and p^k denotes the joint probability of these k vertices to possess exactly k connected vertices, then $\binom{V-1}{k}p^k$ provides the probability of these k vertices to have exactly k another interconnected vertices. However, in this analysis, it should be imposed that there are no more edges beyond these k , i.e., for the reminiscent quantity of vertices, $V - 1 - k$, the complementary probabilistic event of existing edges, that is, $(1 - p)^{(V-1-k)}$, must happen. In view of this reasoning, the degree distribution follows a Binomial distribution with parameters $\text{Binomial}(V - 1, p)$, whose equation is governed by the following expression:

$$P(k) = \binom{V-1}{k} p^k (1-p)^{(V-1)-k}. \quad (2.2)$$

Given that $V \rightarrow \infty$ and $p \ll 1$, one can show that a Binomial distribution parameterized with $\text{Binomial}(N-1, p)$ asymptotically approximates a Poisson distribution with parameter $\text{Poisson}(\lambda)$ ([Meyn and Tweedie, 2009](#)), with the following linking condition:

$$(N-1)p = \lambda. \quad (2.3)$$

Recapitulating from the probability theory that the mean, μ , and the variance, σ^2 , of a $\text{Poisson}(\lambda)$ are given by $\mu = \sigma^2 = \lambda$, one can note that, from the observation of the network in Figure [2.1b](#), which has been built with the parameters $V = 6\,000$ e $p = 0.01$ using a Binomial distribution, the aforementioned distribution really approximates the Poisson distribution with mean (peak) around $\lambda = (N-1)p = (6\,000-1)0.01 \approx 60$.

Moreover, the average shortest path $\langle l \rangle$ is small in these types of network. This quantity increases proportionally to the logarithm of the network size, i.e., $\langle l \rangle \sim \frac{\ln(N)}{\ln(\langle k \rangle)}$, where $\langle k \rangle$ is given by the mean value of the Poisson distribution (mean degree), meaning that $\langle k \rangle = \lambda = (N-1)p$, provided that $V \rightarrow \infty$ and $p \ll 1$ ([Costa et al., 2007](#)).

The big discovery of Erdős and Rényi was that many important properties of a random network may be unveiled as one modifies the parameters of a $\text{Binomial}(N-1, p)$, i.e., for values of the connecting probability p larger than a critical probability p_c , almost all random networks present a specific property Q with probability 1, while the random networks do not present the property if $p \leq p_c$. For example, if p is larger than a certain value of p_c , the random networks can present a single connected component. But, for values below this critical threshold, the random networks no longer present a single component, but instead several unconnected subgraphs. Many other interesting properties have been discussed in the literature and many of them are visited in [Newman \(2003b\)](#).

Small-World Networks

Many real-world networks exhibit the small-world property, i.e., most vertices can be reached by other by means of a small number of intermediate steps (edges). This characteristic is found, for example, in social networks, where virtually everyone in the world can be reached by a short chain of people ([Watts, 2003; Watts and Strogatz, 1998](#)).

In order to build a network that presents the small-world property, one can use the

following network formation process:

- Initially, the network is regular, comprising V vertices, as shown in the left-most network in Fig. 2.2, in which each vertex connects to its k nearest neighbors in each direction, totalizing $2k$ connections;
- Then, each edge is randomly relocated, i.e., given an arbitrary vertex $i \in \mathcal{V}$, we randomly choose one of its original $2k$ connections. The selected edge, say linking vertices i and $j \in \mathcal{V}$, is randomly relocated, such that the destination from j is switched to another vertex $u \in \mathcal{V}, j \neq u$, with probability p .

When $p = 0$, no rearrangements are performed and, therefore, the network will continue to be regular. Conversely, when $p \rightarrow 1$, all edges are effectively relocated (Watts and Strogatz, 1998). Figure 2.2 brings a schematic of the behavior of the parameter p , responsible for the relocation frequency of the edges. Note that, for small values of p , the resulting network is virtually a regular one. As p increases (but still remains small), the property of small-world becomes apparent. When $p = 1$, the network turns out to be random. In this case, the peak of the degree distribution, following this approach, is situated close to $2k$ (Watts, 2003; Watts and Strogatz, 1998).

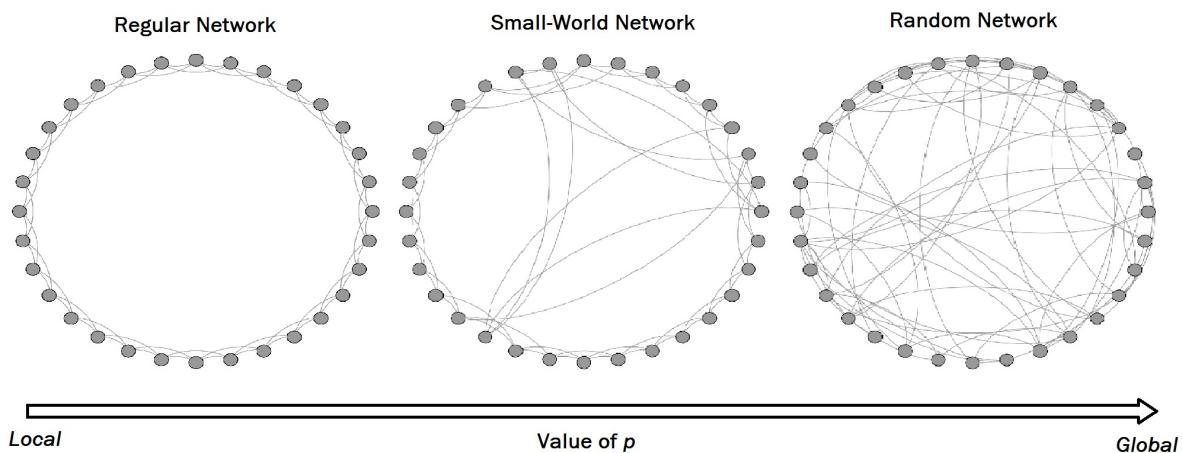


Figure 2.2: Network behavior as we increase the parameter p , which is responsible for the relocation frequency of the edges.

The immediate implication for networks that have the property of small world is that the transport of any information, given that it was generated at any arbitrary vertex of the network, is very fast. For example, the viral contagion: given that a person has contracted some virus, which is living in an environment conducive to their multiplication and spread, then it is expected that, in a short time, many people will be infected by this virus because of its rapid transportation over these types of networks.

Scale-Free Networks

In a study conducted by Albert and Barabási, they noticed that some networks have a small number of vertices with large degree, while most of them have very small degrees ([Barabási and Albert, 1999](#)). With this observation in mind, in 1999, they proposed a new type of network denominated scale-free networks, in which the distribution degree obeys a power-law, as follows:

$$P(k) \sim k^{-\gamma}, \quad (2.4)$$

where γ is a scaling exponent. Analytically, it can be shown that this network model, whose procedure is going to be studied in the following, evolves into a scale-invariant state, which follows a power law with exponent $\gamma = 3$, regardless of the initial set of vertices ([Barabási and Albert, 1999](#)). Note that, by setting a fixed value for γ , as the degree k grows, the number of vertices having degree k decreases. Thus, it is expected that $P(k)$ will have a large value for low values of k and a small value for large values of k , which is consistent with the observation found by Barabási and Albert.

Figure 2.3 shows an illustrative network which shares the scale-free properties. Note that there are very few vertices with high degree, while the great majority (terminal vertices) has low degree.

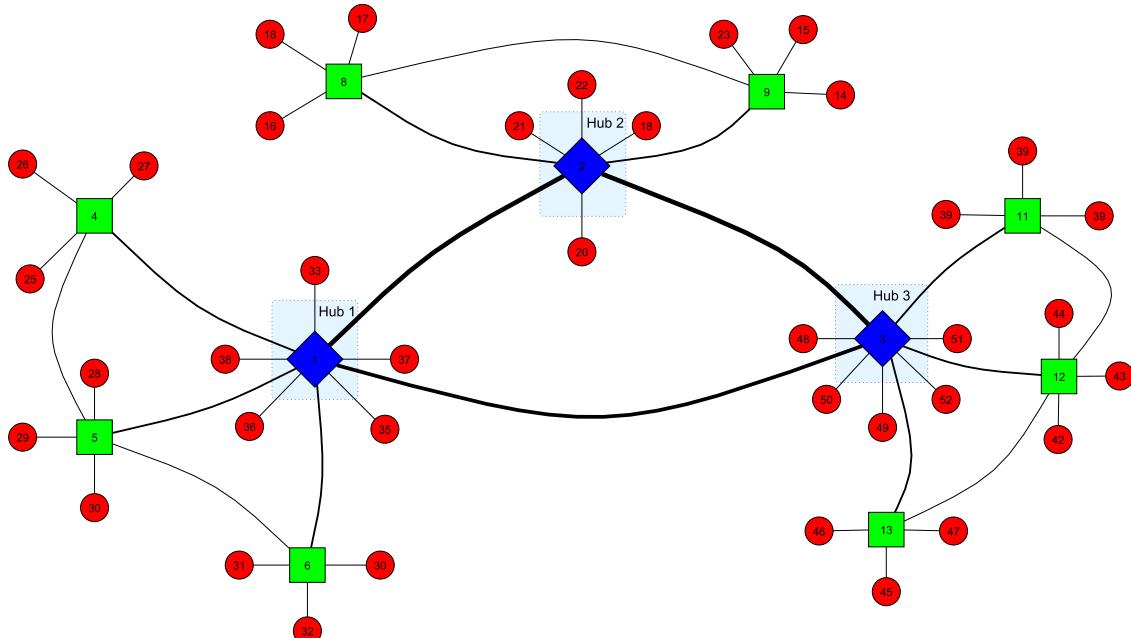


Figure 2.3: Schematic of a scale-free network. The hubs (vertices with high degrees) have been evidenced. Note that there are very few vertices with high degrees, while the great majority (terminal vertices) has low degrees.

Random Clustered Networks

Some real-world networks, such as social and biological ones, present modular structures ([Girvan and Newman, 2002](#)). These networks or communities consist of sets of vertices that satisfy a simple rule: vertices belonging to the same community have many interconnecting edges, while vertices of different communities have few edges interconnecting the other communities. A model for generating such communities was proposed by [Girvan and Newman \(2002\)](#). This agglomerative method groups V initially isolated vertices into M communities. This is managed by creating a link between two vertices with probability p_{in} , if they belong to the same community, or with probability p_{out} , if they belong to distinct communities. The values for p_{in} and p_{out} can be arbitrarily chosen to control the number of intracommunity and intercommunity links, z_{in} and z_{out} , respectively, for an arbitrary average network degree $\langle k \rangle$.

A high value of p_{in} and a low value of p_{out} refer to a network with well-defined communities, i.e., there is a high concentration of edges confined within each community and very few edges interconnecting different communities. Conversely, a low value of p_{in} and a high value of p_{out} contribute to the appearance of communities highly connected with each other, i.e., they become highly mixed. On the basis of these parameters, we are able to define the fraction of intracommunity links $z_{\text{in}}/\langle k \rangle$ and, likewise, the fraction of intercommunity links $z_{\text{out}}/\langle k \rangle$. The quantity $z_{\text{out}}/\langle k \rangle$ defines the mixture of the communities, i.e., as $z_{\text{out}}/\langle k \rangle$ increases, the communities become more mixed and harder to be identified. As we will further see in Section 2.1.4, these quantities are usually employed to compare different competing techniques using the Girvan-Newman's benchmark, which adopts the random clustered networks discussed here.

Empirically, $p_{\text{out}} \ll p_{\text{in}}$ must be satisfied in order to guarantee the presence of communities in the network. Figure 2.4 illustrates a network with four well-defined communities. Observe that the communities in this figure are well-defined, since the number of edges connecting the vertices of the same community is much larger than the number of edges interconnecting different communities.

2.1.4 Community Detection

In this section, we provide an overall view of the concepts and techniques revolving around community detection, as well as a brief description of the state-of-art related techniques. In addition, we present some broadly accepted community detection benchmarks.

Relevant Concepts

Modern science related to networks brought a substantial advance in understanding complex networks. One of the features evident and prominent in complex net-

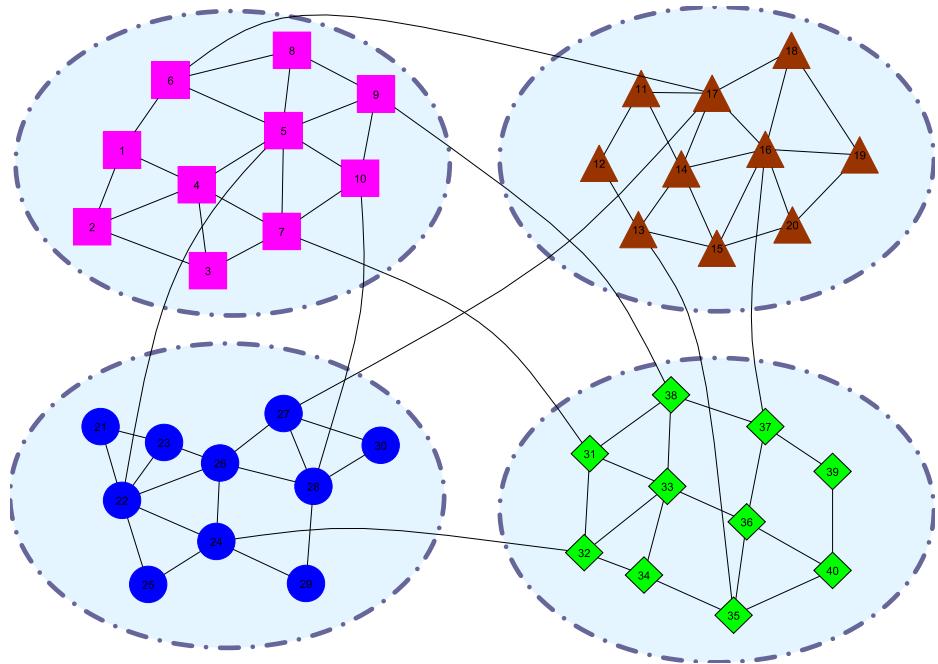


Figure 2.4: Schematic of a random clustered network. Each community is distinguished by a unique color or format.

works is the presence of *communities*. The notion of community is straightforward: each community is defined as a subgraph whose vertices are densely interconnected, and, at the same time, these vertices have few links with the remainder of the network. The community detection in complex networks has become an important topic in graphs and data mining ([Danon et al., 2005](#); [Fortunato, 2010](#); [Newman and Girvan, 2004](#)). In graph theory, the community detection corresponds to the graph partitioning problem, which has been shown to be an NP-complete problem ([Fortunato, 2010](#)).

The study of community detection is very important for the understanding of various phenomena in complex networks ([Gulbahce and Lehmann, 2008](#)). Modular structure introduces important heterogeneities in complex networks. For example, each module can have different local statistics ([Newman, 2006a](#)); some modules may have many connections, while other modules may be sparse. When there is large variation among communities, global values of statistical measures can be misleading. The presence of modular structure may also alter the way in which dynamical processes (e.g. spreading processes and synchronization ([Arenas et al., 2006](#)))) unfold on the network. In biological networks, communities correspond to functional modules in which members of a module function coherently to perform essential cellular tasks. Both metabolic networks ([Ravasz et al., 2002](#)) and protein phosphorylation networks ([Jin and Pawson, 2012](#)), for example, are modular structures.

A promising computational approach to discover functions of genes and proteins is to identify functional modules in biological networks. Since modules are sets of genes or proteins that perform biological processes together, it is possible to classify

proteins with unknown functions by determining what module they belong to (Palla *et al.*, 2005). Correct identification of functional modules also has important biotechnological and drug design applications. In many cases, the deletion of a certain function may be necessary and this can be achieved by removing the entire functional module.

Several distinct ways of detecting modules in complex networks have been proposed (Fortunato, 2010). One popular approach considers communities as sets of adjacent motifs (Palla *et al.*, 2005), other methods are inspired by information theory (Rosvall and Bergstrom, 2007), message passing (Frey and Dueck, 2007), or Bayesian principles (Hofman and Wiggins, 2008; Newman and Leicht, 2007). A widely used class of algorithms is based on the optimization of a quantity called modularity (Newman and Girvan, 2004).

Another important aspect related to community structure is the hierarchical organization displayed by most networked systems in the real world (Fortunato, 2010). Real networks are usually composed of communities including smaller communities, which in turn include smaller communities, and so on. The human body offers a paradigmatic example of hierarchical organization: it is composed by organs, organs are composed by tissues, tissues by cells, etc. Another example is represented by business firms, who are characterized by a pyramidal organization, going from the workers to the president, with intermediate levels corresponding to work groups, departments and management. The generation and evolution of a system organized in interrelated stable subsystems are much quicker than the unstructured system, because it is much easier to assemble the smallest subparts first and use them as building blocks to get larger structures, until the whole system is assembled. In view of these examples, it is clear that the study of community presence in networks plays an important role in understanding various concepts presented in various branches of science.

Another important concept in this field of study is the detection of overlapping vertices (Fortunato, 2010). These vertices are defined as members of more than one community or class at the same time (Palla *et al.*, 2005). For example, in a network of semantic association concepts (Kiss *et al.*, 1973), the term “Brilliant” may be a member of several classes, such as the one representing the concepts related to “Light,” to “Astronomy,” “Color,” and so on (Palla *et al.*, 2005). In a social network, each person naturally belongs to the company where he/she works and also to the group representing the members of his/her family. Given this scenario, the discovery of overlapping vertices and communities is important for data analysis in general.

Overview of the State-of-Art Techniques

Given that the task of accurately solving a problem of community detection is NP-complete, many efforts have been expended towards the development of approximate and efficient solutions. Some of these include the spectral method (Newman, 2006b),

the betweenness-based technique (Newman and Girvan, 2004), modularity optimization (Newman, 2004), detection of communities based on the Potts model (Reichardt and Bornholdt, 2004), synchronization (Arenas *et al.*, 2006), information theory (Fortunato *et al.*, 2004), and random walks (Zhou, 2003a). It is worth citing that Fortunato (2010) presents a recent review on this topic.

Regarding the techniques which aim at detecting overlapping vertices and communities, various methods have been proposed in the literature (Evans and Lambiotte, 2009; Lancichinetti *et al.*, 2009; Nicosia *et al.*, 2009; Palla *et al.*, 2005; Shen *et al.*, 2009; Sun *et al.*, 2011; Zhang *et al.*, 2007). In Zhang *et al.* (2007), the authors combine the idea of the modularity function Q , spectral relaxation, and fuzzy C-Means clustering method in order to build a new modularity function based on a generalized Newman and Girvan's Q function, which is an approximation mapping of the network's vertices into the Euclidean space. In Palla *et al.* (2005), the community structure is uncovered by means of a k -clique percolation and the overlaps among communities are guaranteed by the fact that one vertex can participate in more than one clique. However, the k -clique percolation method gives rise to an incomplete cover of the network, i.e., some vertices may not belong to any community. In addition, the hierarchical structure may not be revealed for a given k . In Lancichinetti *et al.* (2009), it is presented an algorithm that concomitantly finds both overlapping communities and the hierarchical structure based on a fitness function and a resolution parameter. Recently, Evans and Lambiotte (2009) proposed a method to recognize the overlapping community structure by partitioning a graph built from the original network. A salient drawback of the majority of these techniques resides in the fact that the detection of the overlapping characteristics of the input network is performed as a separated or dedicated process apart from the standard community detection technique. In this way, additional computational time is required. As a result, the whole process may have high computational complexity.

In the Section 2.2.2, where we deal with Machine Learning , we will again talk about community detection when dealing with network-based unsupervised techniques.

Community Detection Benchmarks

In this section, we will discuss about two community detection benchmarks, which are frequently used for comparing different competing techniques. In Chapter 3, we will employ both types of benchmarks with the goal of comparing different algorithms.

Girvan-Newman's Benchmark: The Girvan-Newman's benchmark has been proposed by (Girvan and Newman, 2002). This agglomerative method groups V initially isolated vertices into M communities. This is managed by creating a link between two vertices with probability p_{in} , if they belong to the same community, or with probability p_{out} , if they belong to distinct communities. The values of p_{in} and p_{out} can be

arbitrarily chosen to control the number of intracommunity and intercommunity links, z_{in} and z_{out} , respectively, for an arbitrary average network degree $\langle k \rangle$. On the basis of these parameters, we are able to define the fraction of intracommunity links $z_{in}/\langle k \rangle$ and, likewise, the fraction of intercommunity links $z_{out}/\langle k \rangle$. The quantity $z_{out}/\langle k \rangle$ defines the mixture of the communities, i.e., as $z_{out}/\langle k \rangle$ increases, the communities become more mixed and harder to be identified.

The benchmark works by varying the mixture of the communities, i.e., $z_{out}/\langle k \rangle$, for a fixed network comprising V vertices and M communities. For each run, the community detection accuracy is registered. After all the runs have been properly performed, a curve is formed on a two-dimensional plot. This curve serves the purpose of comparing the community detection performance of a control algorithm in relation to competing techniques.

Benchmark of Lancichinatti et al.: The Girvan-Newman's benchmark suffers from several caveats, among them we can list: (i) all vertices of the network have essentially the same degree, (ii) the communities are of the same size, and (iii) the network is small. Motivated by the fact that real-world networks are characterized by heterogeneous distributions of vertex degree, whose tails often decay as power laws, the authors in ([Lancichinetti et al., 2008](#)) have recently proposed a benchmark that generates networks such as to overcome the mentioned problems. The constructed networks assume that both degree and community size distributions follow a power law function, with exponents γ and β , respectively. Typical values of real-world networks are: $2 \leq \gamma \leq 3$ and $1 \leq \beta \leq 2$. Moreover, a mixing parameter μ is employed to interconnect communities in the following manner: each vertex shares a fraction $1 - \mu$ of its links with the other vertices of the same community and a fraction μ with the vertices of other communities.

The benchmark process consists in varying the mixing parameter μ and evaluating the normalized mutual information index, which is a similarity measure of partitions borrowed from the information theory ([Danon et al., 2005](#)) that measures the mutual dependence of different random variables.

2.1.5 Complex Network Measures

In this section, we review some measures that have been proposed in the complex networks literature. Special attention is going to be given for some of these measures, since they will be used in our project later on, specifically for the definition of the hybrid classifier, introduced in Chapter 5.

Assortativity

The assortativity measure numerically translates the preference for vertices of a network to attach to others that are similar or different regarding the vertex's degree in a structural sense (Newman, 2003a). Assortativity is often operationalized as a correlation among vertices. The assortativity coefficient r is essentially the Pearson correlation coefficient of degree between pairs of linked vertices. Hence, positive values of r indicate a correlation between vertices of similar degree, while negative values indicate relationships between vertices of different degrees (Newman, 2002). In general, r lies between -1 and 1 . When $r = 1$, the network is said to have perfect assortative mixing patterns, while at $r = -1$ the network is completely disassortative.

Many studies have been conducted and some conclusions have been drawn on some types of real-world networks. For example, social networks often have apparent assortative mixing. On the other hand, the technological and biological networks frequently appear to be disassortative (Newman, 2002).

The assortativity may be calculated for each existing component j of the network, which possesses a subset of edges \mathcal{U}_j . Consider that u represents an edge, i_u, k_u indicate the degrees of the vertices at each end of the edge u . Also, suppose that there are $E = |\mathcal{E}|$ links in the network. Then, the assortativity of component j is given by:

$$r^{(j)} = \frac{E^{-1} \sum_{u \in \mathcal{U}_j} i_u k_u - \left[E^{-1} \sum_{u \in \mathcal{U}_j} \frac{1}{2} (i_u + k_u) \right]^2}{E^{-1} \sum_{u \in \mathcal{U}_j} \frac{1}{2} (i_u^2 + k_u^2) - \left[E^{-1} \sum_{u \in \mathcal{U}_j} \frac{1}{2} (i_u + k_u) \right]^2}. \quad (2.5)$$

Clustering Coefficient

The clustering coefficient measure quantifies the degree to which local vertices in a network tend to cluster together. Evidence suggests that in many real-world networks, and in particular social networks, vertices tend to create tightly knit groups characterized by a relatively high density of ties (Watts and Strogatz, 1998). Several generalizations and adaptations of such measure have been proposed in the literature (Latapy et al., 2008; Opsahl and Panzarasa, 2009). Here, we define the measure originally proposed by Watts and Strogatz (Watts and Strogatz, 1998). The local clustering coefficient of a vertex in a graph quantifies how close its neighbors are to being a clique (complete graph). Mathematically speaking, the local clustering coefficient of vertex i that is member of the component j is given by:

$$CC_i^{(j)} = \frac{2|e_{uk}|}{k_i^{(j)} (k_i^{(j)} - 1)}, \quad (2.6)$$

where $|e_{uk}|$ the number of links shared by the direct neighbors of vertex i (number of triangles formed by vertex i and any of its two neighbors) and $k_i^{(j)}$ is the degree of vertex i of component j . By (2.6), we see that $CC_i^{(j)} \in [0, 1]$. Having calculated the local clustering coefficient of all vertices that belong to the component j , we are able to define the component's average clustering coefficient as:

$$CC^{(j)} = \frac{1}{n_j} \sum_{i=1}^{n_j} CC_i^{(j)}, \quad (2.7)$$

where n_j symbolizes the number of vertices in component j and $CC^{(j)} \in [0, 1]$. Roughly speaking, the clustering coefficient tells how well-connected the neighborhood of the vertex is. If the neighborhood is fully connected, the clustering coefficient is 1 and a value close to 0 means that there are hardly any triangular connections in the neighborhood.

Average Degree

The degree of a vertex is defined as the number of links emanating from it and the average network degree is the average value of the degrees of the vertices in a given network. The average degree of an arbitrary component representing class j is given by:

$$\langle k^{(j)} \rangle = \frac{1}{n_j} \sum_{i=1}^{n_j} k_i^{(j)}, \quad (2.8)$$

where $k_i^{(j)}$ indicates the degree of vertex i of the component j .

Betweenness

The betweenness, considered as a centrality-based measure, measures the extent to which a vertex lies on the shortest paths between every pair of vertices in a network (Freeman, 1977, 2004; Newman, 2010). Suppose we have a network in which the vertices exchange messages among themselves. Let us initially make the simple assumption that every pair of vertices in the network exchanges a message with equal probability per unit time and that messages always take the shortest (geodesic) path of the network, or one of such paths, chosen at random, if there are several. Then, let us ask the following question: if we wait a suitably long time until many messages have passed between each pair of vertices, how many messages, on average, will have passed through each vertex *en route* to their destination? The answer is that, since

messages are passing down each geodesic path at the same rate, the number passing through each vertex is simply proportional to the number of geodesic paths the vertex lies on (Newman, 2010). This number of geodesic paths is what it is called betweenness index.

Given this definition, it follows that vertices with high betweenness may have considerable influence within a network by virtue of their control ability over information passing between others. The vertices with the highest betweenness in our message-passing scenario are the ones through which the largest number of messages pass, and if those vertices get to see the messages in question as they pass, or if they get paid for passing the messages along, they could derive a lot of power from their position within the network. The vertices with the highest betweenness are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths taken by messages. In real-world situations, of course, not all vertices exchange communications with the same frequency, and in most cases, communications do not always take the shortest path, due to, for example, political or physical reasons.

Mathematically, let η_{st}^i be 1 if vertex i lies on the geodesic path from s to t and 0 if it does not or if there is no such path (because s and t lie in different components of the network). Then the betweenness centrality x_i is given by (Newman, 2010):

$$x_i = \sum_{s,t \in \mathcal{V}} \eta_{st}^i. \quad (2.9)$$

Perhaps, the greatest drawback of this measure is its computational complexity, since one must find all the geodesic paths between every pair of vertices in a network.

Modularity

The modularity measure quantifies how good a particular division of a network is (Clauset *et al.*, 2004; Newman, 2006a) and was designed to measure the strength of division of a network into modules (also called groups, clusters or communities). For networks with the presence of communities, it ranges from 0, representing total randomness of the network, to 1, which indicates completely separated communities of the network. Mathematically, it is described as:

$$Q = \frac{1}{2E} \sum_{i,j} \left(e_{i,j} - \frac{k_i k_j}{2E} \right) \delta(c_i, c_j) , \quad (2.10)$$

where E represents the total number of edges in the network; k_i stands for the degree

of the vertex i ; $\delta(x, y)$ indicates the Kronecker's Delta, which produces 1 if $x = y$ and 0, otherwise; and e_{ij} characterizes the fraction of edges that join vertices in community i to j .

The main idea of modularity is to calculate the fraction of edges that fall within the given groups minus the expected value if edges were distributed at random. For a given division of the network's vertices into some modules, modularity reflects the concentration of vertices within modules compared with a random distribution of links between all vertices, regardless of modules.

Tourist Walks

A tourist walk can be conceptualized as a walker (tourist) aiming at visiting sites (data items) in a d -dimensional map, representing the data set. At each discrete time step, the tourist follows a simple deterministic rule: it visits the nearest site which has not been visited in the previous μ steps. In other words, the walker performs partially self-avoiding deterministic walks over the data set, where this self-avoiding factor is limited to the memory window $\mu - 1$. This quantity can be understood as a repulsive force emanating from the sites in this memory window, which prevents the walker from visiting them in this interval (refractory time). Therefore, it is prohibited that a trajectory to intersect itself inside this memory window. In spite of being a simple rule, it has been shown that this kind of movement possesses complex behavior when $\mu > 1$ ([Lima et al., 2001](#)).

The tourist's behavior heavily depends on the data set's configuration and the starting site. In computational terms, the tourist's movements are entirely realized by means of a neighborhood table. This table is constructed by ordering all the data items in relation to a specific site. This procedure is performed for every site of the data set.

Each tourist walk can be decomposed in two terms: (i) the initial *transient part* of length t and (ii) a *cycle* (attractor) with period c . Figure 2.5 shows an illustration of a tourist walk with $\mu = 1$. In this case, one can see that the transient length is $t = 3$ and the cycle length, $c = 6$.

Considering the attractor or cycle period as a walk section that begins and ends at the same site of the data set may lead one to think that, once the tourist visits a specific site, a new visit to it would configure an attractor. Nevertheless, this is a very simple, and likely to fail, approach for attractors' detection. In fact, during a walk, a site may be re-visited without configuring an attractor. Besides, the tourist's finite memory μ allows some steps of the walk to be repeated without configuring an attractor. For instance, if we had chosen a $\mu = 6$ for the walk in Fig. 2.5, the re-visit performed by the tourist on the site 4 would have not configured an attractor, since the site 5 would still be forbidden to be visited again; hence, the tourist would be compelled to visit another site. This characteristic enables sophisticated trajectories over the data set, at

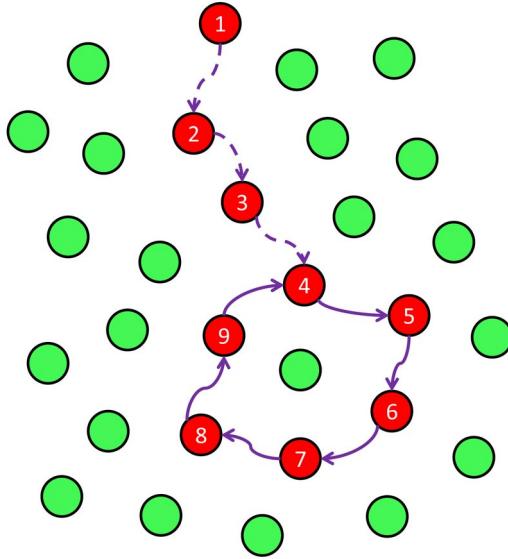


Figure 2.5: Illustration of a tourist walk with $\mu = 1$. The red (dark gray) and green (light gray) dots represent visited and unvisited sites, respectively. The dashed lines indicate the transient part of the walk, whereas the continuous lines, the attractor of the walk.

cost of also increasing the difficulty of detecting an attractor.

In the majority of the works related to tourist walk ([Kinouchi et al., 2002](#); [Lima et al., 2001](#); [Stanley and Buldyrev, 2001](#)), the tourist may visit any other site other than the ones contained in its memory window. As μ increases, there is a significant chance that the walker will begin performing large jumps in the data set, since the neighborhood is most likely to be already visited in its entirety within the time frame μ . As we will show in this thesis, in the context of classification, this is an undesirable characteristic that can be simply avoided by using a graph representation of the input data. In this way, the walker is only permitted to visit vertices, represented now by the sites, that are in its connected neighborhood (link). With this modified mechanism, it is most probable that, for large values of μ , depending on the network configuration, the walker will get trapped within a vertex, not being able to further visit other vertices of the neighborhood. In this scenario, we say that the walk only had a transient part and the cycle period is null ($c = 0$).

2.2 Machine Learning

The second main topic of this thesis is Machine Learning, which will be presented here, with the focus on network-based methods. Firstly, some basic concepts are provided. Following that, we present the three main paradigms of the machine learning area: unsupervised, semisupervised, and supervised learning. For each of them, we discuss the state-of-art network-based techniques, supplying in-depth detail for the most representative techniques, for the sake of completeness.

2.2.1 Overview of Machine Learning

Machine learning aims at developing computational methods that are capable of “learning” with accumulated experiences (Bishop, 2007; Duda *et al.*, 2000; Mitchell, 1997). Traditionally, there are two fundamental types of learning in machine learning. The first is entitled *unsupervised learning*, whose main task consists in revealing the intrinsic structure embedded in the data. The learning process, in this case, is guided by the provided data, since no prior knowledge about the existing classes is required (Mitchell, 1997). Fundamentally, it can be said that the problem of unsupervised learning consists in estimating the density function which generated the data distribution under analysis (Chapelle *et al.*, 2006). Among the main tasks of unsupervised learning, one can highlight: clustering (Girvan and Newman, 2002; Karypis *et al.*, 1999; Newman, 2006a,b), outlier detection (Liu *et al.*, 2004; Lu *et al.*, 2003), dimensionality reduction (Lim and Park, 2009), association (Piatetsky-Shapiro, 1991), among many others. In a clustering task, we expect to find groups in which data items in the same group are very similar to each other, while data items pertaining to different groups are expected to be dissimilar. In this case, the resemblance or similarity of different data items is judged according to an adopted similarity function (Mitchell, 1997). In outlier detection, the goal is to find data items that differ, at a large extent, from the majority of the other data items, i.e., from the original generated distribution (Liu *et al.*, 2004). In dimensionality reduction, the objective is to dispose the data items over a lower dimensional space in relation to its original distribution, so as to simplify the relationships among the data items (Lim and Park, 2009). In association, one seeks to generate rules that relate subsets of predictive attributes (Piatetsky-Shapiro, 1991).

The second type of learning is referred to as *supervised learning*, whose objective is to deduce concepts regarding the data, based on the presented labeled instances, which is commonly denoted as the training set. With this respect, the learning process tries to construct a mapping function conditioned to the provided training set. Often, the learners are tested using unseen data items that compose the so-called test set. When the labels comprise discrete values, then the problem is denominated *classification*, whereas when the values are continuous, *regression* (Bishop, 2007).

The main difference of both learning paradigms, i.e., supervised and unsupervised learning, lies in the fact that the first explores the external information of the training set, which is available at the training stage, in order to induce the hypothesis of the classifier. For example, in a classification task, this external information is materialized in the learning process by the form and values of the labels. In this case, the goal is to create a predictive function that can successfully generalize from this training set, when applied to unknown data (test set). On the other hand, unsupervised learning seeks behaviors or trends in the data, trying to group them in a way that similar data

tend to be agglomerated together and dissimilar data tend to stay on distinct groups. Note that, in this case, no external information is used.

Besides these two well-defined areas, a new machine learning area has recently received attention from the researchers, which is denominated *semisupervised learning*. This new paradigm has been proposed in order to combine the strengths of each learning paradigm. In a typical semisupervised classification, few data are labeled, while most of them are unlabeled. The goal is to propagate the labels from the labeled examples to the unlabeled examples. Therefore, a semisupervised task utilizes both information from the training and the test sets to make predictions.

For didactic purposes, Fig. 2.6a shows a clustering task in unsupervised learning. Note that no external information is available *a priori*, and the clustering is performed by using similarity or topological information of the data. In Fig. 2.6b, it is illustrated a semisupervised classification in semisupervised learning. Note now that some data already possess labels (external information) beforehand, while most of them do not have. The classifier will propagate these labels to the remaining unlabeled data. Finally, in Fig. 2.6c, a supervised classification in supervised learning is displayed. Initially, the classifier is trained by solely using information from the training set, which is always fully labeled. In the next phase, called classification phase, the classifier is used to predict class labels of unseen data items.

2.2.2 Network-based Unsupervised Learning

Unsupervised learning methods are guided exclusively by the intrinsic structure of the data items throughout the learning process, i.e., without any sort of external knowledge. In the following, we present such paradigm with the focus on network-based approaches.

Mathematical Formalization and Fundamental Assumptions

The unsupervised learning can be defined as follows (Alpaydin, 2004; Bishop, 2007; Gan, 2007; Jain *et al.*, 1999; Mitchell, 1997). Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a data set, where n is the total number of data items involved in the learning process. Then, x_i is a d -dimensional vector, where each of the entries is called a feature or descriptor, which has the role to qualitatively or quantitatively describe the data item. Typically, it is assumed that the points are independently and identically distributed in accordance with a common distribution. In the unsupervised learning, no professors or external sources are used, i.e., no labels are provided throughout the learning process. Therefore, it is valid to say that no training phase is involved in this type of learning. As a consequence, the algorithm must be guided by the data distribution itself to infer useful knowledge or trends. For example, in the data clustering or community detection

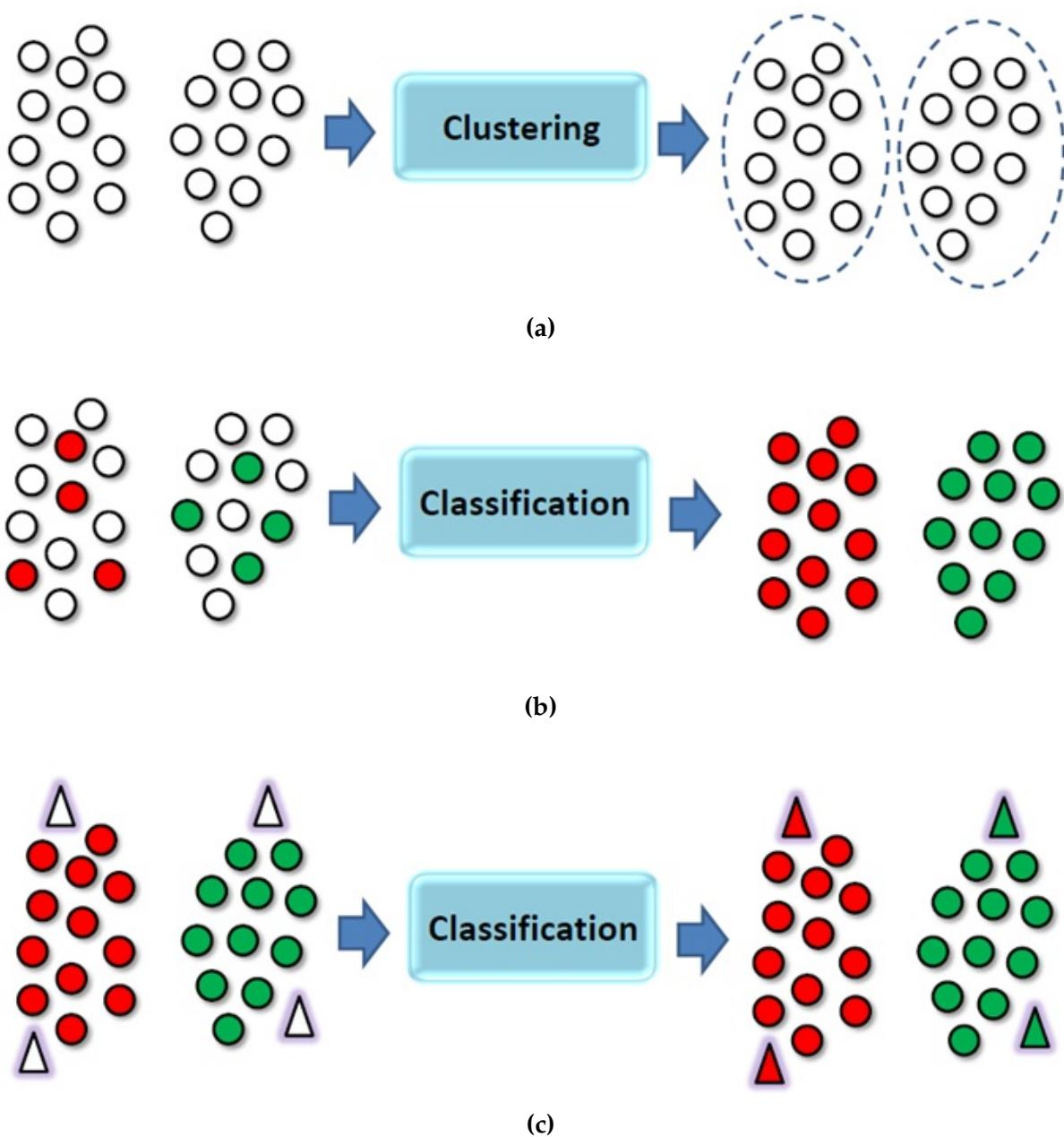


Figure 2.6: Schematic of the three paradigms of the machine learning area. (a) unsupervised learning (a clustering task is presented in the figure); (b) semisupervised learning (a semisupervised classification task is displayed in the figure); and (c) supervised learning (a supervised classification task is depicted in the figure).

tasks, the objective is to find subgroups of data items or vertices, such that the constituents or members of the same subgroup are more similar than different groups, i.e., one seeks groups $\{\mathcal{X}_1, \dots, \mathcal{X}_k\}$, in such a way that $\cup_{i=1}^k \mathcal{X}_i = \mathcal{X}$.

Clustering results differ according to the assumptions made about the data. Some of these forms include (Bishop, 2007; Gan, 2007):

- *The process, which generated the data, has some analytical form:* the methods are biased towards finding clusters with a pre-defined form (similar with estimation). For example, K-Means seeks circular-shaped clusters (MacQueen, 1967). This bias will surely be reflected on the final results yielded by the classifier and will limit its power of detection.
- *Neighbors have the same category:* the algorithms rely on local information to infer their decisions. For example, the k -NN clustering decides in accordance with the neighborhood (Hastie *et al.*, 2009).
- *Data are produced in some “hierarchical” organization:* the techniques that fall into this category are the hierarchical techniques, including the divisive and agglomerative ones. For practical methods of the former, the community detection based on continuously removing edges with the maximum betweenness (Newman and Girvan, 2004) and the bisecting K-Means (Kashef and Kamel, 2009) are representative examples. As good examples of the latter, one can cite the simple-link (Jain *et al.*, 1999; Mitchell, 1997), complete-link (Alpaydin, 2004; Jain *et al.*, 1999), and modularity greedy optimization (Clauset *et al.*, 2004).
- *Data fall into clusters according to some preferred directions:* the methods of this type deal with the transformation of the original data into a modified space, usually with a different number of dimensions. Examples include the principal component analysis (PCA) (Jolliffe, 2002), independent component analysis (ICA) (Jolliffe, 2002), and spectral graph algorithms (Chung, 1997).

Overview of the Techniques

One of the most common tasks of unsupervised learning is data clustering. Formally, data clustering aims at discovering the natural groups of a set of patterns, points, or objects by means of a similarity measure (Bishop, 2007; Gan, 2007; Jain, 2010; Jain *et al.*, 1999; Xu and II, 2005). Each cluster is a collection of data items which are similar between them and are dissimilar to the objects belonging to other clusters. Data clustering is vital in several exploratory pattern-analysis, grouping, decision-making, and machine-learning situations. Some of them include data mining, document retrieval, image segmentation, bioinformatics, and pattern classification (Cinque *et al.*,

2004; Duda *et al.*, 2001; Husek *et al.*, 2006; Jain, 2010; Jain *et al.*, 1999). Unfortunately, in the majority of such tasks, only a little prior information is available about the data. In this way, advances in the methodology to automatically understand, process, and summarize the data are required. Nowadays, this becomes even more critical by virtue of the exponential increase in both the volume and the variety of data (Jain, 2010; Xu and II, 2005). In this scenario, the decision-maker must perform as few assumptions about the data as possible. It is under these practical restrictions that the clustering procedure is specially appropriate for the exploration of inter-relationships among the data points to make an assessment (perhaps preliminary) of their structure (Jain, 2010; Jain *et al.*, 1999). Data clustering algorithms are generally divided in two types: hierarchical or partitional (Berkhin, 2002; Duda *et al.*, 2001; Jain, 2010). The former finds successive clusters using previously established clusters, whereas the latter determines all clusters at once. Hierarchical algorithms can be agglomerative (“bottom-up”) or divisive (“top-down”). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Two-way clustering, co-clustering or bi-clustering are the names for clusterings where not only the objects are clustered but also the features of the objects, i.e., if the data is represented in a data matrix, the row and columns are clustered simultaneously (Bishop, 2007; Gan, 2007; Xu and II, 2005). In both approaches, the algorithms may be further categorized in network-based algorithms or non network-based algorithms.

In relation to partitional methods, several techniques have been studied in the literature. Without doubt, the most well-known one and the pioneer in the field is the K-Means method (MacQueen, 1967). Even though it suffers from several caveats, such as the strong dependence of the performance of the algorithm on the initial conditions of the system and the inherent bias to find only circular-shaped clusters, it has been further enhanced and studied by the community until today. Some methods, which have improved on the basic idea of K-Means, have been proposed, such as the K-Medoids and Fuzzy C-Means (Alpaydin, 2004; Gan, 2007; Jain *et al.*, 1999).

With regard to hierarchical methods, the most well-known techniques are the single-link and the complete-link (Alpaydin, 2004). Among other traditional techniques, we can also include average-Link and the Ward methods (Mitchell, 1997). These algorithms differ in the way that the similarity of the groups is calculated. For example, in the single-link, the similarity is established by finding the minimum distance between every pair of vertices of two any given groups. On the other hand, the complete-link technique finds the similarity based on the maximum distance of two examples of different clusters, rather than the minimum (Gan, 2007).

In general, hierarchical algorithms are commonly more robust than partitional-based algorithms (Jain *et al.*, 1999). For instance, the single-link method is able to find

groups very apart from each other, concentric groups, and chain-like groups that the K-Means technique would have trouble with. However, hierarchical algorithms often take more time in order to process the data items, which may invalidate their employment in large-scale data sets.

Network-based Approaches

In this section, we shift our attention to network-based unsupervised methods. In essence, data clustering can be considered as a community detection problem once a network is constructed from the original data set. In this case, each vertex corresponds to a data item and the connections are established by using a certain similarity measure. It is worth remembering that, in the Section 2.1.4, the notion of communities has already been introduced. Here, we focus on state-of-art techniques.

The clusters in a community detection task are often denominated communities. As we have seen, the notion of community in networks is straightforward: each community is defined as a subgraph whose vertices are densely connected within itself, but sparsely connected with the rest of the network. Community detection in complex networks has turned out to be an important topic in graph mining and data mining (Danon *et al.*, 2005; Fortunato, 2010; Newman and Girvan, 2004). In graph theory, community detection corresponds to graph partition, which has been shown to be a NP-complete problem (Fortunato, 2010). For this reason, a lot of efforts has been spent to develop efficient but suboptimal solutions, such as the spectral method (Newman, 2006b), the technique based on the “betweenness” measure (Newman and Girvan, 2004), modularity optimization (Newman, 2004), community detection based on the Potts model (Reichardt and Bornholdt, 2004), synchronization (Arenas *et al.*, 2006), information theory (Fortunato *et al.*, 2004), and random walks (Zhou, 2003a). For a recent review of this topic, see (Fortunato, 2010).

In the following, we present two representative techniques of this paradigm. Specifically, a special insight is given for the technique proposed in Quiles *et al.* (2008), since, in this project, we will develop new techniques that, in essence, are inspired by this method.

1. Community Detection using Particle Competition: The technique proposed by Quiles *et al.* (2008) has served as an inspiration for one of the main developments in this work. In general, this technique relies on the competition among the particles in a networked environment. These particles navigate in the network with the purpose of dominating new vertices, while also trying to defend their previous dominated territory.

A particle, denoted by ρ_j , is mathematically expressed by two scalar variables: (i) $\rho_j^v(t)$ - which represents the vertex v_i being currently visited by particle ρ_j at time t -

and (ii) $\rho_j^\omega(t) \in [\omega_{\min}, \omega_{\max}]$ - which indicates the exploration potential of particle ρ_j at time t . The update rules that govern the movement and the exploration potentials of the particles are given by:

$$\rho_j^v(t+1) = v_i, \quad (2.11)$$

$$\rho_j^\omega(t+1) = \begin{cases} \rho_j^\omega(t) & \text{if } v_i^\rho(t) = 0 \\ \rho_j^\omega(t) + (\omega_{\max} - \rho_j^\omega(t))\Delta_\rho & \text{if } v_i^\rho(t) = \rho_j \neq 0 \\ \rho_j^\omega(t) - (\rho_j^\omega(t) - \omega_{\min})\Delta_\rho & \text{if } v_i^\rho(t) \neq \rho_j \neq 0 \end{cases}, \quad (2.12)$$

where Δ_ρ controls the exploration level variation that each particle gains or loses, depending on the nature of the vertex which it visits. Specifically, if it visits an already dominated vertex, then the particle's exploration level is strengthened; otherwise, it is decremented.

Each vertex v_i in the network is represented by three scalar variables: (i) $v_i^\rho(t)$, which defines the proprietary particle of the vertex v_i at time t ; (ii) $v_i^\omega(t)$, which indicates the level of domination imposed by particle ρ_j on vertex v_i at time t ; and (iii) $v_i^\gamma(t)$, which symbolizes whether the vertex v_i is being visited by any of the particles at time t . With the help of these variables, the dynamical behavior of the vertices is governed by the following set of equations:

$$v_i^\rho(t+1) = \begin{cases} v_i^\rho(t) & \text{if } v_i^\gamma(t) = 0 \\ \rho_j & \text{if } v_i^\gamma(t) = 1 \text{ and } v_i^\omega(t) = \omega_{\min} \end{cases}, \quad (2.13)$$

$$v_i^\omega(t+1) = \begin{cases} v_i^\omega(t) & \text{if } v_i^\gamma(t) = 0 \\ \max\{\omega_{\min}, v_i^\omega(t) - \Delta_v\} & \text{if } v_i^\gamma(t) = 1 \text{ and } v_i^\rho(t) \neq \rho_j \\ \rho_j^\omega(t+1) & \text{if } v_i^\gamma(t) = 1 \text{ and } v_i^\rho(t) = \rho_j \end{cases}, \quad (2.14)$$

where Δ_v denotes the exploration level fraction lost by a vertex, if a rival particle visits it.

The detection algorithm begins by putting K particles into random vertices. At the beginning of the dynamical process, each particle ρ_j and each vertex v_i have their potentials set to $\rho_j^\omega(0) = \omega_{\min}$ and $v_i^\omega(t) = \omega_{\min}$, respectively. At each iteration, each particle travels to a neighboring vertex, in accordance with a movement policy consisted in a combination of deterministic and random walks. In the former, the particle

randomly visits the neighbors of the currently visited vertex, while, in the latter, the particle prefers to visit vertices that are being dominated by the same particle. In the following, we illustrate the cases which may occur when a particle is on the process of choosing the next vertex to visit:

1. If the visited vertex v_i does not belong to a particle: $v_i^\rho(t) = 0$, the vertex starts to be dominated by the visiting particle, i.e., $v_i^\rho(t) = \rho_j$. The particle's potential ρ_j is not altered and the vertex's potential v_i receives the particle's potential: $v_i^\omega(t) = \rho_j^\omega(t)$;
2. If the visited vertex is dominated by the same particle, the visiting particle's potential, ρ_j , is incremented and v_i receives the new potential of the particle: $v_i^\omega(t) = \rho_j^\omega(t)$;
3. If the visited vertex belongs to a rival particle, then the particle's and the vertex's potentials are weakened. If the particle's potential $\rho_j^\omega(t)$ reaches a value lower than ω_{\min} , then this particle is reset to a new randomly chosen vertex. If the potential of the vertex $v_j^\omega(t)$ reaches a value lower than ω_{\min} , then the vertex becomes no longer owned by the previous particle, i.e., it regresses to the free, non-dominated state: $v_j^\omega(t) = 0$.

Thus, the vertex's level of domination increases if it is visited by the same particle that dominates it at the present moment. On the other hand, during the visit of a rival particle, the domination level imposed by the current dominating particle on that vertex is weakened. If this domination is not strong enough, the particle loses its domination over that vertex. In an extensive period of time, it is expected that each particle will dominate a community in the network.

The model proposed in [Quiles et al. \(2008\)](#) has two salient features: (i) high community detection rates and (ii) low computational complexity. However, in its original work, only a procedure of particle competition is effectively introduced, without any formal definition. This precludes any further analyses or predictions on the model's behavior. As it will be seen later on, one of the main contributions of this research is to present a rigorous model of particle competition via a stochastic competitive dynamical system.

2. Modularity Greedy Algorithm: This algorithm has been proposed by ([Clauset et al., 2004](#)). In its essence, the method attempts to maximize the modularity, whose definition has been given in Section [2.1.5](#), by utilizing a greedy strategy at each time step.

Specifically, the matrix responsible for quantifying the potential increment of the modularity is denoted by ΔQ , whose mathematical expression is given by:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2E} - \frac{k_i k_j}{(2E)^2}, & \text{if } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}, \quad (2.15)$$

where ΔQ_{ij} indicates the corresponding increment in the network modularity, in the case that the communities i and j are chosen to be merged at the current time step.

In the algorithm proposed by ([Clauset et al., 2004](#)), at each step, two communities, say i and j , are merged, in such a way that it causes the biggest increment (or the least decrement) of the modularity at a particular step. In the initial configuration, each vertex is a community itself. If one wants to stop the merges when the network configuration reaches its maximum modularity, one can use the stop criterion as follows: once a negative increment is encountered in this greedy process, the maximum global value associated to the modularity has been reached and subsequent merges will only monotonically decrease the modularity of the network. Therefore, by looking at the signal of ΔQ_{ij} at each iteration is sufficient to know when to stop merging. In addition, no restrictions on the communities to be merged are specified by the original model.

A major advantage of the proposed method is its nonparametric nature: no parameter tuning phase is necessary. Moreover, it has been shown that the modularity greedy algorithm always reaches its maximum value by the procedure that we have described. As we have seen, once this value is reached, subsequent merges will only decrease the modularity value. However, it has been shown that modularity suffers from a resolution limit and, therefore, it is unable to detect very small communities ([Arenas et al., 2008](#); [Fortunato and Barthélémy, 2007](#); [Kumpula et al., 2007](#); [Lancichinetti and Fortunato, 2011](#)). Roughly speaking, the modularity compares the number of edges inside a cluster with the expected number of edges that one would find in the cluster if the network were a random network with the same number of vertices and where each vertex keeps its degree, but edges are otherwise randomly attached. This random null model implicitly assumes that each vertex can get attached to any other vertex of the network. Such assumption is however unreasonable if the network is very large, as the horizon of a vertex includes a small part of the network, ignoring most of it. Moreover, this implies that the expected number of edges between two groups of vertices decreases if the size of the network increases. So, if a network is large enough, the expected number of edges between two groups of vertices in the modularity's null model may be smaller than one. If this happens, a single edge between the two clusters would be interpreted by modularity as a sign of a strong correlation between the two clusters, and optimizing modularity would lead to the merge of the two clusters, independently

of the clusters' features. So, even weakly interconnected complete graphs, which have the highest possible density of internal edges, and represent the best identifiable communities, would be merged by the modularity optimization process if the network is sufficiently large. For this reason, optimizing modularity in large networks would fail to resolve small communities, even when they are well defined. This bias is inevitable for methods like modularity optimization, which rely on a global null model.

2.2.3 Network-based Supervised Learning

Algorithms that exclusively utilize external information in order to induce their hypotheses are considered supervised learning methods. In this section, we deal with supervised learning, with a focus on network-based techniques.

Mathematical Formalization and Fundamental Assumptions

The mathematical formulation of a supervised learning task is reported as follows ([Bishop, 2007](#); [Mitchell, 1997](#)). Consider that it is given a training set denoted here as $\mathcal{X}_{\text{training}} = \{(x_1, y_1), \dots, (x_l, y_l)\}$, where the first component of the i th tuple $x_i = (f_1, \dots, f_d)$ denotes the attributes of the d -dimensional i th training instance. The second component $y_i \in \mathcal{L} = \{L_1, \dots, L_L\}$ characterizes the class label or target associated to that training instance. The goal here is to learn a mapping $x \mapsto y$. Usually, the constructed classifier is checked by using a test set $\mathcal{X}_{\text{test}} = \{x_{l+1}, \dots, x_{l+u}\}$, in which labels are not provided. In this case, each data item is called test instance. For an unbiased learning, the training and test sets must be disjoint, i.e., $\mathcal{X}_{\text{training}} \cap \mathcal{X}_{\text{test}} = \emptyset$. Usually, $n = l + u$ denotes the total number of data items in the learning process.

In the supervised learning scheme, there are two phases of learning: the *training phase* and the *classification phase* ([Bishop, 2007](#)). In the training phase, the classifier is induced or trained by using the training instances (labeled data) in $\mathcal{X}_{\text{training}}$. In the classification phase, the labels of the test instances in $\mathcal{X}_{\text{test}}$ are predicted using the induced classifier.

In order to the classifier to generalize well, some assumptions are often required for the data sets under analysis in the process of learning. Some of them include ([Bishop, 2007](#); [Mitchell, 1997](#)):

- The test set may not be biased towards the training set in order to this estimation to be valid, albeit must be sampled from the same distribution that generated the data from the training set. This assumption makes clear that, since the classifier has been trained in accordance with the distribution of the training set, it is fair enough that it will be capable of efficiently inferring unseen examples of that same distribution. In practice, however, this assumption is often violated to

a certain degree. Strong violations will clearly result in poor classification rate accuracies;

- The training set must be a representative sample in relation to the distribution or population that generated it. Since the hypothesis that the classifier induces is based upon the training set, then if it is not a representative sample of the distribution, the classifier will probably be deceived and, hence, predict in accordance with another distribution, possibly different from the real one, which satisfies the biased training set.

Overview of the Techniques

The techniques are divided into the following groups:

- *Decision trees*: A decision tree consists of vertices and branches that serve the purpose of breaking a set of samples into a set of covering decision rules. In each vertex, in its original inception, a single test or decision is made to obtain a partition. The starting vertex is usually referred to as the root vertex. In the terminal vertices or leaves, a decision is made on the class assignment. In each vertex, the main task is to select an attribute that makes the best partition between the classes of the samples in the training set ([Quinlan, 1992](#));
- *Rule induction*: One of the most expressive and human readable representations for learned hypothesis is sets of IF-THEN rules. In these kinds of rules, the IF part contains conjunctions and disjunctions of conditions composed by the predictive attributes of the learning task, and the THEN part contains the predicted class for the samples that satisfy the IF clause ([Quinlan, 1987](#));
- *Artificial Neural Networks*: Neural networks are interconnected groups of neurons that use a mathematical or computational model for information processing based on a connectionistic approach. In most cases, an artificial neural network is an adaptive system that changes its structure based on external or internal information that flows through the network. In more practical terms, neural networks are nonlinear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. The neural net, ignorant at the start, will, through a “learning” process, become a model of the dependencies between the descriptive variables and the behavior to be explained. The key part in developing neural nets is the architecture that will be used (how many layers, thresholds utilized by the neurons, etc.) and the learning algorithm (back-propagation, etc.) ([Haykin, 2008](#));

- *Bayesian networks*: Bayesian networks constitute a probabilistic framework for reasoning under uncertainty. From an informal perspective, bayesian networks are directed acyclic graphs, where the vertices are the random variables and the edges specify the dependence assumptions that must be held between the random variables. Bayesian networks are based upon the concept of conditional independence among variables. Once the network is constructed, it is used as an efficient device to perform probabilistic inference. This probabilistic reasoning inside the net can be carried out by exact methods, as well as by approximate methods ([Korb and Nicholson, 2010](#)). A special case of bayesian networks is when no dependences on the predictive variables exist. In this case, the classifier is best known as Naïve Bayes ([Neapolitan, 2003](#));
- *Statistical Learning Theory*: Maybe the most well-known technique of this type of learning is the Support Vector Machines (SVM), which is based on the principle of structural risk minimization. Originally, it was worked out for linear two-class classification with margins, where margin means the minimal distance from the separating hyperplane to the closest data points. SVM learning machine seeks for an optimal separating hyperplane, where the margin is maximal. An important and unique feature of this approach is that the solution is based only on those data points that are at the margin. These points are called support vectors. The linear SVM can be extended to a nonlinear one when first the problem is transformed into a feature space using a set of nonlinear basis functions. In the feature space - which can be very high dimensional - the data points can be separated linearly. An important advantage of the SVM is that it is not necessary to implement this transformation and to determine the separating hyperplane in the possibly very-high dimensional feature space, instead a kernel representation can be used, where the solution is written as a weighted sum of the values of a certain kernel function evaluated at the support vectors ([Vapnik, 1995](#));
- *Instance-based learning*: Instance-based learning has its root in the study of the Nearest Neighbor algorithm. The simplest form of nearest neighbor or, more generally, k -Nearest Neighbor (k -NN) algorithms, simply stores the training instances and classifies a new instance by predicting that it has the same class as its nearest stored instance or the majority class of its k nearest stored instances, according to some distance measure. The essence of this learning method resides in the form of the similarity function that computes the distances from the new instance to the training instances ([Cover and Hart, 1967](#));
- *Network-based methods*: The inference is done by means of a network constructed from the training set. Up to now, there are still few network-based supervised

learning techniques ([Hasan et al., 2006](#)). A detailed study will be conducted in the next section.

Network-based Approaches

Network-based unsupervised and semisupervised learning techniques have been extensively studied in the literature ([Chapelle et al., 2006](#); [Jain, 2010](#)). However, there are still few reported network-based supervised learning techniques ([Bertini Junior et al., 2011](#)). With this respect, there is still a great space for the proposal and discovery of new ways of supervised learning in networked environments. Presumably, several network-based semisupervised inductive methods, such as those presented in Refs. ([Belkin et al., 2006](#); [Chen et al., 2009](#); [Sindhwani et al., 2005](#)) can be converted into a supervised learning scheme when a reasonable number of labeled instances is provided. However, these methods are aimed at considering unlabeled instances during the training phase, and a graph-based approach is employed to model the data into a manifold, in order to first propagate the labels to all unlabeled instances. Thus, if the majority of instances in the data set is labeled, a regular supervised approach, using labeled instances only, would be preferable ([Bertini Junior et al., 2011](#)).

Another type of network-based classification approach refers to relational classification. Such type of data differs from typical data because it violates the instance-independence assumption, which means that the class label of an instance might not depend only on its own attributes, but also on the labels of its neighbors ([Macskassy and Provost, 2003](#)). This kind of data is usually presented in a graph form (also termed within-network data) with some of the vertices labeled and the rest unlabeled. The task is to infer the labels for the unlabeled vertices. Relational classification techniques can be applied to solve a wide range of problems, for example, discovering molecular pathways in a gene expression ([Segal et al., 2003](#)) and classification of linked scientific research papers ([Lu and Getoor, 2003](#)), link prediction ([Liben-Nowell and Kleinberg, 2007](#)), among others. For example, in link prediction on social networks ([Barabási et al., 2002](#); [Dorogovtsev and Mendes, 2003](#); [Hasan et al., 2006](#); [Liben-Nowell and Kleinberg, 2007](#)), the specific problem is to predict the likelihood of a future association between two vertices, knowing that there is no association between the vertices in the current state of the graph ([Hasan et al., 2006](#)). The problem has a wide variety of further applications: recommender systems, identification of probable professional or academic associations in scientific collaboration networks or e-commerce sites, identification of structures of criminal networks and structural analysis in the field of microbiology or biomedicine etc. All these demand for much more efficient and versatile approaches for link prediction and thereby making it an important and scientifically attractive research topic. Another application that is also related to relational classification is defined as the detection of small connection subgraphs that best capture the relationship

between two vertices in a social network ([Faloutsos et al., 2004](#)). In the same referenced work, the authors also proposed an efficient algorithm based on electrical circuit laws to find the connection subgraph from large social networks. It has also been shown that a connection subgraph can be used to effectively compute several topological feature values for the supervised link prediction problem, especially when the network is very large ([Hasan et al., 2006](#)).

State-of-the-art approaches to spread labels along the graph consider inferring the labels of interrelated vertices in a jointly manner is known as collective inference ([McDowell et al., 2009](#)). This kind of inference can significantly reduce classification error when compared to traditional inference techniques ([Jensen et al., 2004](#)). Collective inference methods may use both data attributes and data relational features to perform classification. Traditionally, vector-based methods have treated data items as independent, which makes it possible to infer class membership on an instance per instance basis. With networked data, the class membership of one data item (vertex) may have an influence on the class membership of a related vertex. Furthermore, vertices not directly linked may be related by chains of links, which suggests that it may be beneficial to infer the class memberships of all vertices simultaneously. Collective inferencing in relational data makes simultaneous statistical judgments regarding the values of an attribute or attributes for multiple entities in a graph for which some attribute values are not known ([McDowell et al., 2009](#)).

In the literature, some algorithms have been proposed that may only employ collective inference on specific phases of the learning process. For example, one may employ a local classifier, such as Naïve Bayes or relational probability trees, to predict labels for each unlabeled vertex and further use a collective inference algorithm, such as ICA ([Lu and Getoor, 2003](#)) or Gibbs sampling ([Jensen et al., 2004](#)), to restate the class labels to all the vertices to be used at the next iteration. Such kinds of methods are called local classifier-based methods. Another kind of approach, called global formulation-based methods, does not use a separate local classifier, but it uses the entire algorithm for the training and inference, also using relational and non-relational data. Such approach conducts training with the objective of optimizing a global objective function. Examples of these algorithms include loopy belief propagation and relaxation labeling ([Sen et al., 2008](#)). In search of a unification on relational data classification by means of utilizing networks, [Macskassy and Provost \(2007\)](#) proposed a general framework. The framework builds a model considering three components: a local classifier inducer, which makes use of a training set to estimate probability distribution over the classes; a relational classifier, which also aims to estimate a probability distribution but now considering the neighboring relations in the graph; and a collective inference component, which further refines the class predictions.

While collective inference presents some advantages, in some cases, inferring la-

bels collectively leads to uncertainties that may actually lead to lower classification accuracies when compared to non-relational approaches. For example, an incorrectly predicted label may influence the predictions of its neighbors in future iterations, possibly cascading this error through long chains of vertices (McDowell *et al.*, 2009). On one hand, there is a tendency to represent data by networks; on the other hand, some approaches consider transforming network data into raw, vector-based data in order to apply classical methods, such as SVM and neural networks. This kind of method requires extracting features from the networked data in order to construct a trainable vector-based set. The task of feature extraction from a given relational data can be divided according to the presence or absence of labels in the vertices, and named label-dependent and label-independent extraction, respectively. The former uses both graph structure and label information throughout the neighboring vertices and the latter exclusively considers the graph structure (McDowell *et al.*, 2009). An approach to estimate the similarity between edges in a graph or between two graphs as a whole is graph kernel. Briefly, these kinds of methods use a kernel to establish a similarity measure on graphs. The main difficulty in this approach is to define a kernel that is suitable for the graph structure and reasonably efficient to be evaluated.

In the following, we present two representative network-based supervised learning techniques.

1. Classification using k -associated graphs: This technique has recently been proposed by (Bertini Junior *et al.*, 2011). The basis of the k -associated graphs lies on representing the training set as a graph (network), more specifically a directed graph referred to as k -associated graph. Such a graph is built from a vector-based data set by abstracting data items to vertices and similarities to edges. After a k -associated graph is constructed for a given k , the purity measure for every component in the graph is computed and is used to determine the optimal graph for classification, both on the training and the test phases. A k -associated graph is constructed using a modified version of the k -nearest neighbor technique. In the proposed modification, since we are dealing with supervised classification, a specific vertex from a determined class is only permitted to connect with vertices from the same class. This simple rule generates class components in the overall network.

The purity is defined for each component (isolated subgraph) in the graph as follows: given a parameter k , which is used to construct the networks using the k -nearest neighbor technique, then, a vertex can, at most, receive $2k$ connections. In an undirected graph, this is always true because if vertex j is one of the k -nearest neighbors of i , then the reciprocal is true. However, in the technique proposed in (Bertini Junior *et al.*, 2011), the networks are considered as digraphs, i.e., directed graphs. Therefore, it is expected that each vertex will have degrees ranging from k to $2k$. It is under these

circumstances that the purity measure has been defined. It quantifies the proportion of edges that has effectively been created with vertices from the same class over the total number of possible vertices, $2k$. In mathematical terms, the purity ϕ of component α , ϕ_α , is defined as:

$$\phi_\alpha = \frac{D_\alpha}{2k}, \quad (2.16)$$

where D_α denotes the average degree of component α . In general, a purity value close to 1 indicates that a large portion of edges are among vertices in the component, resulting in high compactness of the component, while lower values reveal high mixing between components of different classes. It is for this reason why ϕ_α is called a purity measure for the component α . Normalized purity measure can also be seen as the *a priori* probability of connections within a component. This property is exploited by the classifier to define the classes of the test instances.

In the referred technique, one can note that the parameter k plays a key role in the process of learning in the training phase. By virtue of this, the authors of ([Bertini Júnior et al., 2011](#)) have developed a procedure for estimating the value k for each of the components. It is intuitive that some graphs may have better components than others according to the purity measure. Rarely, a graph obtained by a unique value of k produces the best configuration of vertices into components for a given data set. A single value of k produces components with nearly the same size, therefore structure and purity are restrained to only one possible value of k at a time. Consequently, it would be better to allow multiples values of k to represent the same data space in order to best fit the data regarding to component size and purity. Bearing this in mind, a suggestive idea is to obtain a graph with the best organization of data into components with different and localized k , i.e., each component has its own optimal k . As a result, all components reach their respective highest purity. Such a graph is called the k -associated optimal graph.

For obtaining the optimal k -associated graph, the rationale is to increase k while keeping the best components found so far starting from the 1-associated graph. For each k and component, the purity measure is calculated and it is used to compare between components of different k -associated graphs formed within different values of k . The component with the highest purity value is maintained, while the others are discarded.

Once the optimal k -associated graph has been properly obtained, then the classification phase begins. In this phase, the authors use a Bayes classifier in order to predict. Specifically, the *a priori* probabilities are calculated using a normalized purity value of each of the class components, rather than the traditional size proportions that we

encounter in the literature.

A great potentiality of this technique is its nonparametric nature, which eliminates the step of parameter tuning. However, since it must create a network from a vector-based data set, then its time complexity is at least of the order of $\mathcal{O}(V^2)$.

2. Network Learning Toolkit: This work has been introduced in ([Macskassy and Provost, 2007](#)). This is a within-networked technique, rather than an across-network method. In essence, the toolkit is composed of three terms, each of which capturing different concepts of the data items, as follows:

1. *Non-relational ("local") model:* This component consists of a (learned) model, which uses only local information - namely information about (attributes of) the data items, whose target variable is to be estimated. The local models can be used to generate priors that comprise the initial state for the relational learning and collective inference components. They also can be used as one source of evidence during collective inference. These models typically are produced by traditional machine learning methods;
2. *Relational model:* In contrast to the non-relational component, the relational model makes use of the relations in the network as well as the values of attributes of related entities, possibly through long chains of relations. Relational models also may use local attributes of the data items;
3. *Collective inferencing:* The collective inferencing component determines how the unknown values are estimated together, possibly influencing each other, as described above.

Depending on the choices of each of the three aforementioned components, one can get new kinds of classifiers, some of which are already known in the community. For example, using a Naïve Bayes classifier as the local model, a Naïve Bayes Markov Random Field classifier for the relational model, and relaxation labeling for the inferencing method forms the system used by Chakrabarti *et al.* ([Macskassy and Provost, 2007](#)).

It is worth registering that the collective inferencing component can exploit relational autocorrelation, which is a widely observed characteristic of relational data. This phenomenon may reveal that a variable for one instance is highly correlated with the value of the same variable on another instance. By making inferences about multiple data instances simultaneously, collective inference can significantly reduce classification error in some cases.

The importance of NetKit is threefold: (i) it generalizes several existing methods for classification in networked data, thereby making comparison to existing methods

possible; (ii) it enables the creation and use of many new algorithms by its modularity and extensibility; and (iii) it enables the analysis/comparison of individual components and configurations. These contributions are welcomed by the literature, because, since then, there has been no systematic study of machine learning methods for within-network classification that compares various algorithms on several data sets.

2.2.4 Network-based Semisupervised Learning

Algorithms that are able to learn using only a few labeled examples have aroused the interest of the Artificial Intelligence community. Semisupervised learning aims, among other features, to reduce the work of human expert in the labeling process. This feature is quite interesting especially when the labeling process is expensive and time consuming, for example, in video indexing, classification of audio signals, text categorization, medical diagnostics, genome data, among others ([Chapelle et al., 2006](#)).

Motivations

Semisupervised learning is a relative new area in relation to unsupervised and supervised learning; therefore, it is important to better understand the reason behind its creation. From an engineering standpoint, it is clear that the collection of labeled data is much more intensive and costly in relation to the collection encompassing the unlabeled data. However, the main purpose of semisupervised learning goes way beyond a purely utilitarian tool. So debatable, most natural learning (human and animal) occurs in a semisupervised regime basis. On the world in which we live, living beings are in a constant exposure to a flow of natural stimuli. Such stimuli include unlabeled data that are easily noticeable. For example, in a context of recognition and phonological acquisition, a child is exposed to many acoustic sounds. Many of these sounds are not familiar to the child. A positive feedback by part of another person is the main source of labeled data. In many cases, a small amount of feedback is sufficient to allow the child to master the acoustic-phonetic mapping of any languages ([Belkin et al., 2005, 2006](#)).

The human has ability to learn unsupervised concepts - for example, clusters and categories of objects -, suggesting that unlabeled data could be satisfactorily used for learning natural invariance to form categories and to construct classifiers. In many pattern recognition tasks, humans only have access to a small amount of labeled patterns. Therefore, the success of human learning in environments with little knowledge indubitably happens with the effective use of large sets of unlabeled data to extract information that is useful for generalization purposes. Consequently, if the goal is to know how natural learning is processed, there is a need to think in terms of semisupervised learning ([Belkin et al., 2004, 2006](#)).

Another motivation for studying semisupervised learning is intrinsically linked to improving the accuracy of the models. In a recent study conducted by [Singh et al. \(2008\)](#), it was shown that, by means of a finite sample analysis, if the distribution complexity under consideration is too high to be learned by n labeled data, but it is small enough to be learned by $m \gg n$ unlabeled data, then the semisupervised learning is able to improve the performance of a typical fully supervised task. As an example, consider Fig. 2.7, where the dark circles denote labeled data, while light circles, unlabeled data. Applying a fully supervised algorithm to this problem, most likely the decision boundary would be established in the surroundings of the dotted line. On the other hand, applying a semisupervised learning algorithm, the decision boundary would probably be fixed around the continuous line. In this example, supervised algorithms would not be able to efficiently classify the unlabeled examples. In contrast, semisupervised methods, with the aid of the unlabeled data used in the training process, would output higher accuracy rates. This is exactly what happens in the figure, which confirms that, in this particular occasion, the semisupervised classifier reflects more faithfully the distribution of the classes.

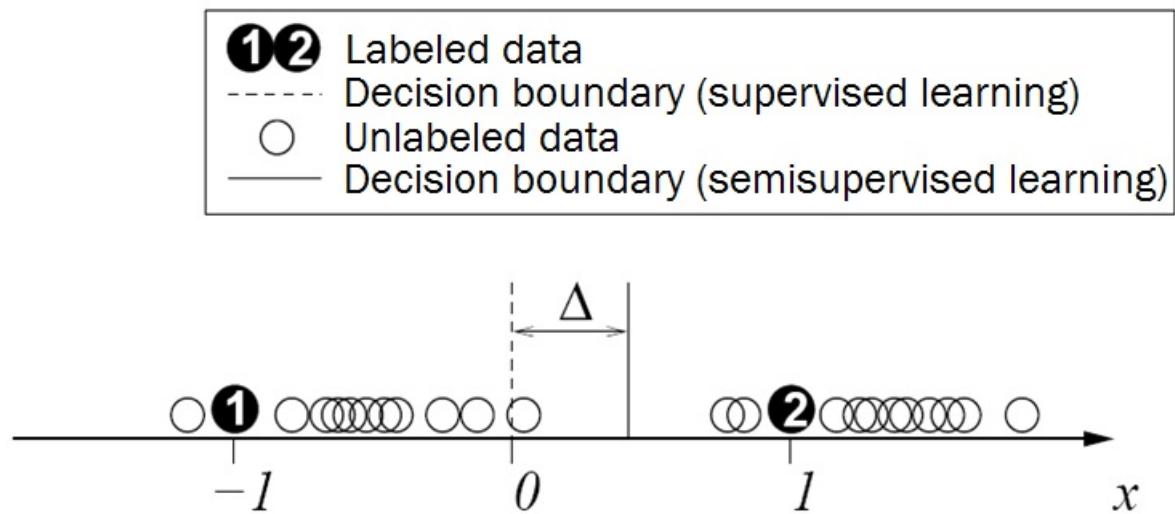


Figure 2.7: An example of data set where semisupervised learning techniques would lead to more robust results than supervised learning techniques. The dotted line denotes the decision boundary that would be probably output by a supervised learning method. The continuous line displays the same information for a semisupervised learning algorithm.

Mathematical Formulation and Hypotheses

Semisupervised learning can be defined as follows ([Chapelle et al., 2006](#)). Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a data set, divided into two parts: $\mathcal{X}_l = \{x_1, x_2, \dots, x_l\}$, where $l < n$, and $\mathcal{X}_u = \{x_{l+1}, \dots, x_n\}$. Consider that $\mathcal{L} = \{L_1, L_2, \dots, L_L\}$ is a set that encompasses all possible label outputs. Since the label set is discrete, this task is referred

to as semisupervised classification (the same reasoning can be applied to regression). The labels of the subset \mathcal{X}_u are not known *a priori*. Normally, $l \ll u$, i.e., the great majority of data items does not possess labels. As we have already stressed, this often happens because the task of manual labeling is cumbersome and often is performed by a human expert. Based on these definitions, semisupervised learning can be used in data classification tasks, when the labeled examples are used in the process of labeling unlabeled examples, but also in clustering tasks. In the latter case, the labeled samples are responsible for guiding the process of forming clusters (Chapelle *et al.*, 2006), subjected to restrictions imposed by the labeled instances.

It is worth mentioning that, for the proper functioning of semisupervised learning techniques, some assumptions about the data consistency are essential (Wang *et al.*, 2008). Typically, a semisupervised learning method relies on one or more of the following assumptions (Chapelle *et al.*, 2006):

- *Cluster Assumption:* Points that belong to the same high-density region, i.e., are located in the same group, are plausible candidates for belonging to the same class;
- *Smoothness Assumption:* Points that are near in the attribute space are probable candidates of being members of the same class. This assumption forces that the decision function yielded by the classifier should be smoother in high-density regions than in locations with low density. This analysis is in line with the cluster assumption and both, hence, complement each other.
- *Manifold Assumption:* This idea is based upon the premise that a set of data in a high-dimension space may be, approximately, reduced to a smaller space (manifold data) via a nonlinear mapping function. This hypothesis is usually employed to soften the curse of dimensionality problem, which is related to the fact that the volume of the space increases exponentially with the number of dimensions, and an exponentially larger number of examples would be needed for constructing induction classifiers with the same accuracy power.

The way that the semisupervised learning algorithms treat these assumptions represents one of the fundamental differences among them.

Overview of the Techniques

Traditionally, the semisupervised learning methods are divided into the following categories:

- *Generative Models:* The inference via generative models involves the estimation of a conditional density. The Expectation Maximization technique is the most

known technique pertaining to this approach ([Nigam et al., 2000](#)). Besides that, a myriad of techniques proposed so far in the literature can be encountered in [Alpaydin \(2004\)](#); [Chapelle et al. \(2006\)](#); [Gärtner \(2008\)](#); [Zhu and Goldberg \(2009\)](#);

- *Cluster-and-label Models*: The inference is done based on the results obtained by a clustering task subjected to some restrictions regarding the pre-labeled data set. Some representative methods are given in [Dara et al. \(2002\)](#); [Demiriz et al. \(1999\)](#).
- *Low Density Region Separation Models*: The inference is based on the development of decision boundary functions, such that each decision boundary is created as far as possible from the high-density regions. Undoubtedly, the most known method of this approach is the Transductive SVM ([Cortes and Vapnik, 1995](#); [Vapnik, 1998](#)). More related techniques can be found in [Alpaydin \(2004\)](#); [Chapelle et al. \(2006\)](#); [Cortes and Vapnik \(1995\)](#); [Zhu and Goldberg \(2009\)](#);
- *Network-based Models*: The inference is done by using a network. This approach is the focus of this thesis and is going to be studied in the next section.

Network-based Approaches

During the last years, the most active area of research in the field of semisupervised learning has been related to methods based on graphs or networks. The common point of the techniques that use this approach lies in the fact that they use the data items as the vertices of the network, while the links between the data items depend on the chosen similarity function and the labels of the vertices ([Chapelle et al., 2006](#)). An advantage of using networks for data analysis is the ability to reveal the topological structure of the data set. For example, consider a semisupervised classification problem, as shown in Fig. 2.8a, in which the distribution of the data has the shape of bananas and only two pre-labeled data items are provided (distinguished by the colors or formats). The outcome of the SVM technique is given in Fig. 2.8b, while the result of the k -nearest neighbor (k -NN) technique is shown in Fig. 2.8c. Both techniques use vector-based data set. The ideal classification, in this case, is given in Fig. 2.8d. As we can see, both algorithms were unable to correctly find the classes. The use of networks in this type of problem may reveal the topological structure of the data, thus allowing the detection of classes and groups with arbitrary shapes ([Karypis et al., 1999](#); [Zhou et al., 2004](#)).

Network-based semisupervised learning begins by constructing a network with the input vector-based data. Once the network is built, the learning process consists in assigning a label for every non-labeled vertex in the test set. The inference is done by using the edges that interconnect the vertices of the network ([Chapelle et al., 2006](#)). It is extremely important to mention that, in contrast to the traditional techniques that make

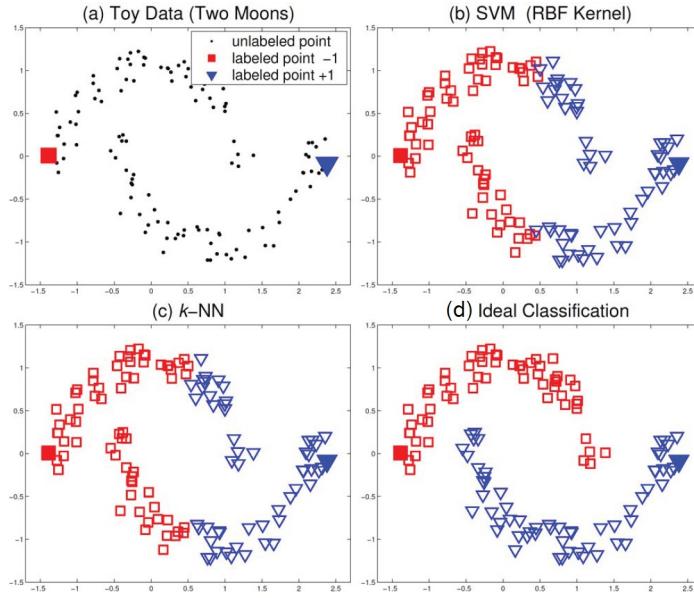


Figure 2.8: Motivation for the research on network-based semisupervised methods. (a) Initial semisupervised classification task. (b) Result obtained by the SVM method. (c) Result achieved by the k -NN technique. (d) Ideal result. This figure has been extracted from [Zhou et al. \(2004\)](#).

use of attribute-value tables to conduct their analyzes on the data, network-based techniques directly use the neighborhood information to analyze the data. As explained in [Zhu \(2005a\)](#), this may generate more robust and efficient classifiers.

In general, network-based methods can be characterized as transductive techniques ([Zhu, 2005b](#)), i.e., the algorithm aims to obtain a value for each non-labeled vertex without the need to generate (induce) a global generalizing function. Among the main advantages of these techniques, one may highlight ([Chapelle et al., 2006; Zhu, 2005b](#)):

- Clusters of various forms can be effectively detected;
- Learning process does not make decisions explicitly based on distance functions;
- Representation of data sets with multiple classes is facilitated;
- Some problems are naturally represented by networks, for example: protein-protein interaction networks, blood mainstream, among others.

Many semisupervised learning techniques, such as Transductive SVM, can identify data classes of well-defined forms, but usually fail to identify classes of irregular forms. Thus, assumptions on the class distributions have to be made ([Chapelle et al., 2006](#)). Unfortunately, such information is usually unknown *a priori*. In order to overcome this problem, graph-based methods have been developed in the last years, like Mincut ([Zhu et al., 2003](#)), Local and Global Consistency ([Zhou et al., 2004](#)), Local Learning Regularization ([Wu and Schölkopf, 2007](#)), Local and Global Regularization

(Wang *et al.*, 2008), Manifold Regularizer (Belkin *et al.*, 2006), Semisupervised Modularity (Silva and Zhao, 2012c), D-Walks (Callut *et al.*, 2008), random walk techniques (Grady, 2006; Szummer and Jaakkola, 2001), and label propagation techniques (Wang and Zhang, 2008; Zhu and Ghahramani, 2002). However, most of the graph-based methods share the same regularization framework, differing only in the particular choice of the loss function and the regularizer (Belkin *et al.*, 2004, 2005; Blum and Chawla, 2001; Joachims, 2003; Zhou *et al.*, 2004; Zhu *et al.*, 2003), and most of them have cubic order of computational complexity ($O(n^3)$). This factor makes their applicability limited to small- or middle-sized data sets (Zhu, 2005a). As data sets get larger and larger, the development of efficient semisupervised learning methods is still necessary.

For the sake of completeness, we discuss two of the aforementioned techniques in detail.

1. Local and Global Consistency: This method has been proposed by Zhou *et al.* (2004) and may be considered as the responsible for leveraging and pioneering the research in network-based semisupervised learning techniques. This method considers the general problem of learning from labeled and unlabeled data by means of the construction of a classification function that is sufficiently smooth with respect to the intrinsic structure that is revealed by both types of data.

Before delving into the specific concepts of such technique, it is valid to register some key points that this technique shares with other methods based on energy minimization. In general, such techniques aim at minimizing a cost expression, consisting essentially of two distinct functions:

1. *Loss Function:* it leads the algorithm to penalize decisions that flip the labels of pre-labeled vertices. Practically, to minimize this term, it is enough to prevent the change of pre-labeled vertices;
2. *Regularizer Function:* it is responsible for modeling the cost of propagating labels to unlabeled vertices. Given that many algorithms rely on the smoothness assumption, this function must be smooth in dense regions of the network.

The technique considers a set of matrices \mathcal{M} with dimensions $n \times L$, all of which with non-negative entries. A matrix $F = [F_1^T, \dots, F_n^T]^T \in \mathcal{M}$ corresponds to the classification of the data items \mathcal{X} , such that, for each unlabeled data item x_i , we designate a label in accordance with the expression $y_i = \arg \max_{j \in \mathcal{L}} F_{ij}$. One can think F as a vectorial function that attributes, for each unlabeled data x_i , the maximum value of $F_j, j \in \mathcal{L}$. In addition, it is defined the matrix Y with dimensions $n \times L$, such that $Y_{ij} = 1$ if x_i is labeled as $y_i = j \in \mathcal{L}$, and $Y_{ij} = 0$, otherwise. The algorithm evolves as follows:

1. Generate the affinity matrix W , which is given by $W_{ij} = \exp\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ if $i \neq j$, and $W_{ii} = 0$, otherwise;
2. Construct the matrix $S = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, in which D is a diagonal matrix with each entry (i, i) equivalent to the sum of the i th row of W ;
3. Iterate $F(t+1) = \alpha SF(t) + (1 - \alpha)Y$ until it converges, where $\alpha \in (0, 1)$;
4. Consider that F^* denotes the limit of the sequence $\{F(t)\}$. Then, label each unlabeled vertex x_i following the formula: $y_i = \arg \max_{j \in \mathcal{L}} F_{ij}^*$.

Moreover, the authors have analytically shown that the sequence $\mathfrak{F} = \{F(t)\}$ converges and assumes the following closed formula:

$$F^* = \lim_{t \rightarrow \infty} F(t) = (I - \alpha S)^{-1} Y. EEE \quad (2.17)$$

Still in Zhou *et al.* (2004), the authors have found a regularization framework that satisfies the aforementioned dynamics. In this framework, one aims at minimizing a cost or energy expression. The encountered expression, here written as $C(F)$, is given as:

$$C(F) = \frac{1}{2} \left(\sum_{i=1, j \in \mathcal{L}}^n W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^n \|F_i - Y_i\|^2 \right), \quad (2.18)$$

where $\mu > 0$ is a regularizer parameter. In this case, the optimal values for the classification function become:

$$F^* = \arg \min_{F \in \mathfrak{F}} C(F). \quad (2.19)$$

The first term in (2.18) enforces a smoothness decision of the classifier, meaning that a good classification function must not have large derivatives in high-density areas. This is exactly the definition of a regularizer function. The second term symbolizes the adjustment restriction, revealing that a good classification function also must not exchange the labels from already labeled data. In this case, this definition perfectly fits into the description of a loss function. The counterweight between these two conflicting quantities is given by the positive constant μ .

The advantage of this technique is its simplicity. As one can see, the propagation is done by utilizing a linear update rule and convergence issues have been fully described, enabling one to understand the dynamics of such model in the long run. However, the algorithm suffers from some drawbacks: since the propagation is done

utilizing a linear function, nonlinear characteristics of the data may pass unseen by the algorithm. Moreover, since a matrix inversion is involved to find the optimal solution, the algorithm requires $\mathcal{O}(V^3)$ to run, which is unfeasible for large-scale networks.

2. Semisupervised Modularity Method: This algorithm has recently been proposed by (Silva and Zhao, 2012c) and is inspired by the modularity greedy algorithm, originally described in (Clauset *et al.*, 2004; Newman, 2006a). In the algorithm proposed in the latter, at each time step, two communities, say i and j , are merged, in such a way that occurs the biggest increment (or least decrement) of the modularity at a particular step. In the initial configuration, each vertex is a community itself. If one wants to stop the merges at the network configuration with maximum modularity, one can use an easy stop criterion as follows: once a negative increment is encountered, the maximum global value associated to the modularity is reached and subsequent merges will monotonically decrease the modularity of the network. No restrictions on the communities to be merged are specified by the original model.

In order to adapt the modularity greedy algorithm for the context of semisupervised learning, we make the following modifications:

- I. Initially, we have l labeled vertices in the network. The task consists of propagating their labels to the unlabeled vertices. Once an unlabeled vertex receives a label, it cannot be changed.
- II. At each step, we merge the communities (at the beginning, each community encompasses only one vertex) in such a way that the modularity is maximized. However, such merging is subjected to some constraints: in light of mimicking the propagation of labels in the network, a merging will only occur if at least one of the communities to be merged has been labeled before. Suppose that communities c_i and c_j have been selected to be merged, each of which carrying the labels c_i^l and c_j^l (let \emptyset denote the no label class), then one of the following four cases occurs:

Case 1. The merging does not occur if $c_i^l \neq c_j^l$, provided that $c_i^l \neq \emptyset$ and $c_j^l \neq \emptyset$. This case represents a clash between two different classes that have been previously labeled.

Case 2. The merging occurs if $c_i^l \neq \emptyset$ and $c_j^l = \emptyset$, or $c_i^l = \emptyset$ and $c_j^l \neq \emptyset$. This case represents the traditional label propagation from a labeled community to an unlabeled one. c_j^l receives the label from c_i^l in the first case and c_i^l receives the label from c_j^l in the second case.

Case 3. The merging occurs if $c_i^l = c_j^l$, provided that $c_i^l \neq \emptyset$ and $c_j^l \neq \emptyset$. In this case, the merging process just puts two communities of the same class together, maximizing the modularity.

Case 4. The merging does not occur if $c_i^l = c_j^l = \emptyset$, since no label is being propagated.

If the merging does not occur, then other two communities, which result in the second highest entry of the ΔQ , are selected to be potentially merged, i.e., the Step II is repeated, and so on, until a valid merging takes place.

Keeping in mind that the modularity algorithm tries to maximize the number of edges among vertices of the same community, while also attempts to minimize the same quantity amongst distinct communities, the dynamics of the proposed procedure will propagate labels so as to maintain the cluster assumptions, such as smoothness and proximity. In this way, the modified modularity greedy algorithm performs the work of propagating the labels in an optimized manner, provided that the network is strongly connected among vertices of the same class and weakly connected among vertices of distinct classes.

For the stop criterion of this algorithm, it simply needs to be run until no unlabeled vertex remains, regardless of the value of the modularity, since we are not looking for a good network division, but an *ordered way* of labeling vertices. The mechanism of maximizing the modularity does this job for us. The convergence is guaranteed to happen and a proof has been provided in ([Silva and Zhao, 2012c](#)).

Additionally, in an attempt to make feasible the semisupervised algorithm to work on large-scale networks, a network reduction technique has also been proposed in the same paper. Let $\varphi(i) \in [0, 1]$ denote the proportion of reduction to be performed over the class pertaining to label $i \in \mathcal{L}$. Then, the reduction is done by the following procedure, step-by-step:

I. Randomly choose pairs of pre-labeled vertices $r \in \Psi_i(t)$ and $s \in \Psi_i(t)$ to coalesce.

In this process, r is removed from the network and s is entitled as a super-vertex, by virtue of the fact that it is representing more than one vertex in the network.

In this process, all the links that are connected to r are redirected to s . Suppose a connection between w and s already exists and w is also a neighbor of r , then we strengthen the connection between w and s by adding the weight from the former edge (w, r) to (w, s) . Figure 2.9 illustrates this idea. This is done until $|\Psi_i(t + \Delta_t)| = (1 - \varphi(i))|\Psi_i(t)|$, where $\Delta_t > 0$ is a parameter bounded by the upper limit provided by the convergence proof previously supplied. Essentially, $|\Psi_i(t + \Delta_t)|$ denotes the size of Ψ_i in a future time that can be reached in a finite number of steps. If $\varphi(i) = 1$, then we deliberately continue to merge until $|\Psi_i(t + \Delta_t)| = 1$, i.e., until only one element remains of that class.

II. All the self-loops, brought into the network by the reduction process, are removed. This prevents the modified modularity greedy algorithm from trying to merge a certain community with itself.

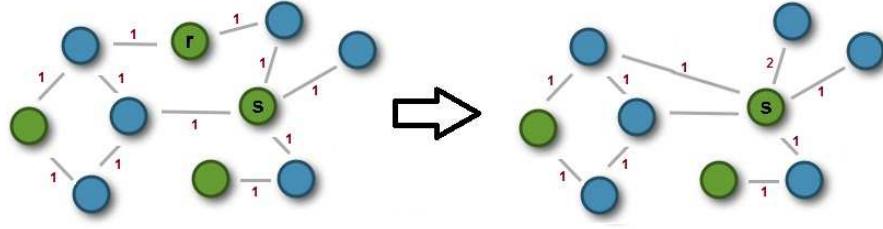


Figure 2.9: Process of coalescence of vertices s and r . After the merge, s consumes r and becomes a super-vertex. All neighbors of r are connected to s during this procedure.

III. This process of reduction is performed for every class $i \in \mathcal{L}$ that exists in the network.

In general, at the end of the reduction process it is expected the network to shrink, because $(V - L) + \sum_{i \in \mathcal{L}} |\Psi_i(t + \Delta_t)| = (V - L) + \sum_{i \in \mathcal{L}} [1 - \varphi(i)]|\Psi_i(t)| \leq V$, since $0 \leq \varphi(i) \leq 1$ and thus $\sum_{i \in \mathcal{L}} [1 - \varphi(i)]|\Psi_i(t)| \leq L$. If the proportion of labeled vertices is high, then this will greatly reduce the network, provided that $\varphi(i), \forall i \in \mathcal{L}$, is high. Usually, the quantity of labeled vertices is low, because the task of labeling is generally expensive and cumbersome.

The main advantage of the aforementioned technique is that it does not require any kinds of parameters in order to work. Considering that the task of parameter tuning often takes a considerable time to complete and that lots of traps are involved in this investigation, such as the presence of local maxima, an expert may be needed such as to set feasible initial kickoff values for the parameters tuning procedure. Undoubtedly, this positive characteristic is very welcomed in a learning process. A major drawback of the aforementioned technique is that it suffers from the inherent resolution problem that the modularity greedy algorithm presents (see Section 2.1.5 for details). This often leads to bad results when there is the presence of classes with very distinct sizes.

2.3 Chapter Remarks

In this chapter, we have reviewed the area of complex networks, showing the advantages inherent to their use, such as the ability to take into account the topological features of the data. Such a feature is important, since it gains independence of the model from particular physical measures, such as similarity functions. In addition, we have reviewed the concepts of community and overlapping vertices detection, as well as some state-of-art techniques. Afterwards, two network-based benchmarks have been discussed that are usually used in the literature for comparison matters. Finally, some important network measures have been presented.

Moreover, we have also delved into the machine learning area, with a focus on network-based techniques. We have first seen a glimpse of the overall organization

that the literature gives for the machine learning. Then, we started to discuss each paradigm of learning in detail, i.e., unsupervised, semisupervised, and supervised learning. We have focused on network-based techniques, because they serve as a natural bridge for the use of complex-network aspects in the learning process. Additionally, we have presented several state-of-art techniques, providing the potentialities and shortcomings of each one of them.

In the next chapters, we will merge together these two macro-areas with the purpose of modeling computational algorithms that infer decisions in networked environments in the three paradigms of learning.

CHAPTER
3

Unsupervised Stochastic Competitive Learning in Complex Networks

This chapter treats the issue of solving unsupervised learning tasks, such as clustering, using complex networks. As we have already drawn attention to, we here present a new type of competitive learning mechanism based on a stochastic nonlinear dynamical system, which runs in a networked environment. Specifically, a set of particles is randomly released into the vertices of a network. As time progresses, they move across the network in accordance with a convex stochastic combination of random and preferential walks, which will be detailed in the further sections. Straightforward applications are in community detection and data clustering. In essence, data clustering can be considered as a community detection problem once a network is constructed from the original data set. In this case, each vertex corresponds to a data item and the connections are established by using a certain similarity measure. Considering such applications, the competitive walking process reaches a dynamics equilibrium when each community or data cluster is dominated by a single particle.

As we will see, the competitive mechanism is formally modeled via a stochastic nonlinear dynamical system. Moreover, empirical and analytical analyses are supplied so as to better enable one to fully understand the potentialities and shortcomings of the model. Given that the models of interactive walking processes correspond to many natural and artificial systems, and due to the relative lack of theory for such systems, we can argue that the analytical analysis provided herein is an important step to understand such systems.

Once the fundamental idea and the model definition are properly presented, we enrich this chapter by providing several applications of the proposed technique in var-

ious interesting problems indicated in the literature. These include the creation of an efficient method for estimating the most likely number of clusters or communities in a data set by using an evaluator index. Such index exploits the domination level information generated by the competition process itself. As a result, if we take into account that the number of clusters is far less than the quantity of data items, the cluster number determination process does not increase the model's complexity order. Since the determination of the actual number of clusters is an important issue in data clustering ([Sugar and James, 2003](#); [Wang et al., 2009](#)), our method also presents a contribution to this topic. Following the same line, an index for detecting overlapping cluster structures is also proposed, which, under some assumptions, may also not increase the model's computational complexity due to its embedded nature within the competitive process. With all these tools at hand, we finalize this chapter by exploiting the application of handwritten digits and letters clustering. Therein, we will verify that the competition process is able to satisfactorily cluster several variations and distortions of the same handwritten digits and letters into their corresponding cluster.

3.1 Model Description

In this section, the proposed competitive learning technique is presented in detail.

3.1.1 A Brief Overview of the Model

Consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \{v_1, \dots, v_V\}$ is the set of vertices and $\mathcal{E} = \{e_1, \dots, e_E\} \subset \mathcal{V} \times \mathcal{V}$ is the set of links (or edges). In the proposed competitive learning model, a set of particles $\mathcal{K} = \{1, \dots, K\}$ is inserted into the vertices of the network in a random manner. Each particle can be conceptualized as a flag carrier with its main objective being to conquer new vertices, while defending its current dominated vertices. In this case, a competition process will naturally take place amongst the particles. When a particle visits an arbitrary vertex, it strengthens its own domination level on that vertex and, simultaneously, weakens the domination levels of all other rival particles on the same vertex. It is expected that this model, in a broad horizon of time, will end up uncovering the communities in the network in such a way that each particle dominates a community.

A particle in this model can be, at any given time, in two possible states: *active* or *exhausted*. Whenever the particle is active, it navigates in the network according to a combined behavior of random and preferential walking. The random walking term is responsible for the adventuring behavior of the particle, i.e., it randomly visits vertices without taking into account their domination levels. The preferential walking term is responsible for the defensive behavior of the particle, i.e., it prefers to reinforce

its owned territory rather than visiting a vertex that is not being dominated by that particle. So as to make this process suitable, each particle carries an energy term with it. This energy increases when the particle is visiting an already dominated vertex by itself, and decreases whenever it visits a vertex that is being owned by a rival particle. If this energy drops under a minimum allowed value, the particle becomes exhausted and is teleported back to a safe ground, which is one of the vertices dominated by the present particle. At the next step, the exhausted particle will be possibly recharged by visiting the vertices dominated by itself. With this confinement mechanism, we expect to restrain the acting region of each particle and, thus, reduce long range and redundant visits in the network.

3.1.2 Deriving the Competitive Transition Matrix of the Dynamical System

In this model, each particle $k \in \mathcal{K}$ can perform two distinct types of movements: (i) a random movement term, modeled by the matrix $\mathbb{P}_{\text{rand}}^{(k)}$, which allows the particle to venture throughout the network, without accounting for the defense of the previously dominated vertices; and (ii) a preferential movement term, modeled by the matrix $\mathbb{P}_{\text{pref}}^{(k)}$, which is responsible for inducing the particle to reinforce the vertices that are owned by itself, i.e., the particle prefers visiting its dominated vertices, instead of a randomly selected one. In order to model such dynamics, consider the stochastic vector $p(t) = [p^{(1)}(t), p^{(2)}(t), \dots, p^{(K)}(t)]$, which denotes the localization of the set of K particles presented to the network. Its k th-entry, $p^{(k)}(t)$, indicates the location of particle k in the network at time t , i.e., $p^{(k)}(t) \in \mathcal{V}, \forall k \in \mathcal{K}$. It is desirable to find a transition matrix that governs the probability distribution of the movement of the particles to the immediate future state: $p(t+1) = [p^{(1)}(t+1), p^{(2)}(t+1), \dots, p^{(K)}(t+1)]$.

With the intent of keeping track of the current states of all particles, we introduce the following stochastic vector: $S(t) = [S^{(1)}(t), \dots, S^{(K)}(t)]$, where the k th-entry, $S^{(k)}(t) \in \{0, 1\}$, indicates whether the particle k is active ($S^{(k)}(t) = 0$) or exhausted ($S^{(k)}(t) = 1$) at time t . When it is active, the movement policy consists of a combined behavior of randomness and preferential movements. When it is exhausted, the particle switches its movement policy to a new transition matrix, here referred to as $\mathbb{P}_{\text{rean}}^{(k)}(t)$. This matrix is responsible for taking the particle back to its dominated territory, in order to reanimate the corresponding particle by recharging its energy. We call this the *reanimation procedure*. After the energy has been properly recharged, the particle can again perform the combined random-preferential movement in the network. In brief, $S(t)$ acts as a switch that determines the movement policy of all particles at time t . With all this information in mind, we are able to define the transition matrix associated to the particle k as:

$$\mathbb{P}_{\text{transition}}^{(k)}(t) \triangleq (1 - S^{(k)}(t)) \left[\lambda \mathbb{P}_{\text{pref}}^{(k)}(t) + (1 - \lambda) \mathbb{P}_{\text{rand}}^{(k)} \right] + S^{(k)}(t) \mathbb{P}_{\text{rean}}^{(k)}(t), \quad (3.1)$$

where $\lambda \in [0, 1]$ indicates the desired fraction of preferential movement that all particles in the network will perform. Specifically, $\mathbb{P}_{\text{transition}}^{(k)}(i, j, t)$ indicates the probability that particle k performs a transition from vertex i to j at time t . It is worth noting that (3.1) is a convex combination of two transition matrices (the first term is itself a combination of two transition matrices, too). Since the sum of the coefficients is unitary, the resulting matrix is guaranteed to be another transition matrix. We now define each matrix that appears in (3.1) in a detailed manner.

The derivation of the random movement matrix is straightforward, since this matrix only depends on the adjacency matrix of the graph, which is previously known. In this way, each entry $(i, j) \in \mathcal{V} \times \mathcal{V}$ of the matrix $\mathbb{P}_{\text{rand}}^{(k)}$ is given by:

$$\mathbb{P}_{\text{rand}}^{(k)}(i, j) \triangleq \frac{a_{i,j}}{\sum_{u=1}^V a_{i,u}}, \quad (3.2)$$

where $a_{i,j}$ denotes the (i, j) th-entry of the adjacency matrix A of the graph. Note that (3.2) resembles the traditional Markovian matrix for a single random walker, here symbolized as a particle (Çinlar, 1975). In addition, note that matrix $\mathbb{P}_{\text{rand}}^{(k)}$ is time-invariant and it is the same for every particle in the network; therefore, whenever the context makes it clear, we drop the superscript k for convenience. In short, the probability of an adjacent neighbor j to be visited from vertex i is proportional to the edge weight linking these two vertices.

In order to assist in the derivation of the matrix associated to the preferential movement term, $\mathbb{P}_{\text{pref}}^{(k)}(t)$, for a given particle $k \in \mathcal{K}$, we introduce the following stochastic vector:

$$N_i(t) \triangleq [N_i^{(1)}(t), N_i^{(2)}(t), \dots, N_i^{(K)}(t)]^T, \quad (3.3)$$

where $\dim(N_i(t)) = K \times 1$, T denotes the transpose operator, and $N_i(t)$ stands for the number of visits received by vertex i up to time t by all particles scattered throughout the network. Specifically, the k th-entry, $N_i^{(k)}(t)$, indicates the number of visits made by particle k to vertex i up to time t . Then, the matrix that contains the number of visits made by every particle in the network to all the vertices is defined as:

$$N(t) \triangleq [N_1(t), N_2(t), \dots, N_V(t)]^T, \quad (3.4)$$

where $\dim(N(t)) = V \times K$. Let us also formally define the domination level vector of vertex i , $\bar{N}_i(t)$, according to the following stochastic vector:

$$\bar{N}_i(t) \triangleq [\bar{N}_i^{(1)}(t), \bar{N}_i^{(2)}(t), \dots, \bar{N}_i^{(K)}(t)]^T, \quad (3.5)$$

where $\dim(\bar{N}_i(t)) = K \times 1$ and $\bar{N}_i(t)$ denotes the relative frequency of visits of all particles in the network to vertex i at time t . In particular, the k th-entry, $\bar{N}_i^{(k)}(t)$, indicates the relative frequency of visits performed by particle k to vertex i at time t . Similarly to the previous case, this notation is extended to all the vertices constituting the network by defining the domination level matrix that sustains all the domination levels imposed by every particle in the network to all the vertices as:

$$\bar{N}(t) \triangleq [\bar{N}_1(t), \bar{N}_2(t), \dots, \bar{N}_V(t)]^T, \quad (3.6)$$

where $\dim(\bar{N}(t)) = V \times K$. Mathematically, each entry of $\bar{N}_i^{(k)}(t)$ is defined as:

$$\bar{N}_i^{(k)}(t) \triangleq \frac{N_i^{(k)}(t)}{\sum_{u=1}^K N_i^{(u)}(t)}. \quad (3.7)$$

In view of this, we can define $\mathbb{P}_{\text{pref}}^{(k)}(i, j, t)$, which is the probability of a single particle k to perform a transition from vertex i to j at time t , using solely the preferential movement term, as follows:

$$\mathbb{P}_{\text{pref}}^{(k)}(i, j, t) \triangleq \frac{a_{i,j} \bar{N}_j^{(k)}(t)}{\sum_{u=1}^V a_{i,u} \bar{N}_u^{(k)}(t)}. \quad (3.8)$$

From (3.8), it can be observed that each particle has a different transition matrix associated to its preferential movement and that, unlike the matrix related to the random movement, it is time-variant with dependence on the domination levels of all the vertices ($\bar{N}(t)$) in the network at time t . It is worth mentioning that the approach taken here to characterize the preferential movement of the particles is defined as the visiting frequency of each particle to a specific vertex. This means that, as more visits are performed by a particle to a determined vertex, there will be a higher chance for the same

particle to repeatedly visit the same vertex. Furthermore, it is important to emphasize that (3.8) produces two distinct features presented by a natural competitive model: (i) the strengthening of the domination level of the visiting particle on a vertex; and (ii) the consequent weakening of the domination levels of all other particles on the same vertex.

For didactic purposes, we now summarize and consolidate the key concepts introduced so far in a simple example given in the following.

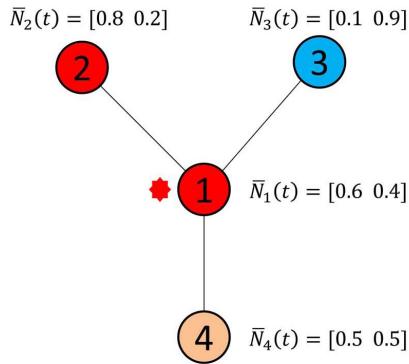


Figure 3.1: A typical situation where the red (dark gray) particle, presently located at vertex 1, has to choose a neighbor to visit in the next iteration. The probability that the red particle will visit a specific vertex is proportional, besides the random term, to the domination level that the same particle imposes on that vertex. For illustration purposes, we have provided the domination level vector for each vertex. In this example, there are two particles, red (dark gray) and blue (gray), which is not shown). The beige (light gray) color denotes vertices that are not being dominated by any particles in the system at time t .

Example 1. Consider the network portrayed in Fig. 3.1, where there are two particles, namely red (dark gray) and blue (gray), and four vertices. For illustration purpose, we only depict the location of the red (dark gray) particle, which is currently visiting vertex 1. In this example, we will make clear how the role of the domination level plays in the determination of the resulting transition probability matrix. Within the figure, we also didactically supply the domination level vector of each vertex at time t . Note that the ownership of the vertex (in the figure, the color of the vertex) is set according to the particle that is imposing the highest domination level on that specific vertex. For instance, in vertex 1, the red (dark gray) particle is imposing a domination of 60%, and the blue (gray) particle, only 40%. Our goal here is to derive the transition matrix of the red particle in agreement with (3.1). Suppose at time t , the red particle is active, therefore, $S^{(\text{red})}(t) = 0$, and, consequently, the second term of the convex combination in (3.1) vanishes. Let $\lambda = 0.8$. On the basis of (3.2), the random movement term of the red particle is given by:

$$\mathbb{P}_{\text{rand}}^{(\text{red})} = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad (3.9)$$

and the preferential movement matrix at the immediate posterior time $t + 1$, according to (3.8), is given by:

$$\mathbb{P}_{\text{pref}}^{(\text{red})}(t + 1) = \begin{bmatrix} 0 & 0.57 & 0.07 & 0.36 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad (3.10)$$

Finally, the transition matrix associated to the red particle is determined by a weighted combination of the random (time-invariant) and the preferential matrices at time $t + 1$, given that the particle is active (see (3.1)). Numerically,

$$\begin{aligned} \mathbb{P}_{\text{transition}}^{(\text{red})}(t + 1) &= 0.2 \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 & 0.57 & 0.07 & 0.36 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0.52 & 0.12 & 0.36 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (3.11)$$

Therefore, the red particle, which is currently in vertex 1, will have a bigger chance to visit vertex 2 (52% chance of visiting) than others. This behavior can be controlled by refining the λ parameter. A high value induces the particle to perform preferential movements, i.e., it will keep visiting its owned vertices in a frequent manner. On the hand, a low value provides a higher weight to the random movement term, making each particle resembles a traditional Markov walker as $\lambda \rightarrow 0$ (Cinlar, 1975). In the extreme case, i.e., $\lambda = 0$, the mechanism of competition is turned off and the model reduces to multiple non-interactive random walks. In this way, our model generalizes the theory of random walks, according to the parameter λ .

Now we define each entry of $\mathbb{P}_{\text{rean}}^{(k)}(t)$ that is responsible for teleporting an ex-

hausted particle $k \in \mathcal{K}$ back to its dominated territory, with the purpose of recharging its energy (reanimation process). Suppose that particle k is visiting vertex i when its energy is completely depleted. In this situation, the particle must regress to an arbitrary vertex j of its possession at time t , according to the following expression:

$$\mathbb{P}_{\text{rean}}^{(k)}(i, j, t) \triangleq \frac{\mathbb{1}_{\left\{ \arg \max_{m \in \mathcal{K}} (\bar{N}_j^{(m)}(t)) = k \right\}}}{\sum_{u=1}^V \mathbb{1}_{\left\{ \arg \max_{m \in \mathcal{K}} (\bar{N}_u^{(m)}(t)) = k \right\}}}, \quad (3.12)$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function that yields 1 if the argument is logically true and 0, otherwise. The operator $\arg \max(\cdot)$ returns an index M , where $\bar{N}_u^{(M)}(t)$ is the maximal value among all $\bar{N}_u^{(m)}(t)$ for $m = 1, 2, \dots, K$. A careful analysis of the expression in (3.12) reveals that the probability of returning to an arbitrary vertex j that is dominated by the particle k follows a uniform distribution. In other words, (3.12) only results in nonzero transition probabilities for vertices j that are being dominated by particle k at time t , regardless of the existence of a connection between i and j in the adjacency matrix. Figure 3.2 illustrates how the reanimation scheme takes place. If no vertex is being dominated by particle k at time t , we deliberately put it in any vertex of the network in a random manner (uniform distribution).

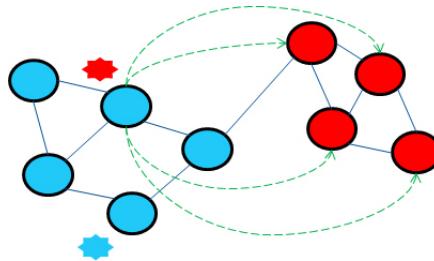


Figure 3.2: Illustration of the reanimation scheme. The red (dark gray) particle has gotten exhausted by virtue of visiting a vertex dominated by a rival particle. As a consequence, its energy has completely depleted. In order to reanimate the particle, it is teleported back to one of its owned vertices. In the next iterations, with a relatively significant probability, its energy will be renewed, since the neighborhood is dominated by the same particle.

Next, the energy update rule of the particles is discussed. First, we define the parameters ω_{\min} and ω_{\max} , which characterize the minimum and maximum energy levels, respectively, that a particle may possess. Moreover, it is useful to introduce the stochastic vector: $E(t) = [E^{(1)}(t), \dots, E^{(K)}(t)]$, where the k th-entry, $E^{(k)}(t) \in [\omega_{\min}, \omega_{\max}]$, $\omega_{\max} \geq \omega_{\min}$, denotes the energy level of particle k at time t . In view of these definitions, the energy update rule is given by:

$$E^{(k)}(t) = \begin{cases} \min(\omega_{\max}, E^{(k)}(t-1) + \Delta), & \text{if } \text{owner}(k, t) \\ \max(\omega_{\min}, E^{(k)}(t-1) - \Delta), & \text{if } \neg \text{owner}(k, t) \end{cases} \quad (3.13)$$

where $\text{owner}(k, t) = \left(\arg \max_{m \in \mathcal{K}} \left(\bar{N}_{p^{(k)}(t)}^{(m)}(t) \right) = k \right)$ is a logical expression that essentially yields true if the vertex that particle k visits at time t (i.e., vertex $p^{(k)}(t)$) is being dominated by it, but yields false otherwise; $\dim(E(t)) = 1 \times K$; $\Delta > 0$ symbolizes the increment or decrement of energy that each particle receives at time t . The first expression in (3.13) represents the increment of the particle's energy and it occurs when particle k visits a vertex $p^{(k)}(t)$ which is dominated by itself, i.e., $\arg \max_{m \in \mathbb{K}} \left(\bar{N}_{p^{(k)}(t)}^{(m)}(t) \right) = k$. Similarly, the second expression in (3.13) indicates the decrement of the particle's energy that happens when it visits a vertex dominated by rival particles. Therefore, in this model, particles will be given a penalty if they are wandering in rival territory, so as to minimize aimless navigation of the particles in the network. With this mechanism, we expect to improve the cluster and community detection rates of the algorithm.

Now we advance to the update rule that governs $S(t)$, which is responsible for determining the movement policy of each particle. As we have stated, an arbitrary particle k will be transported back to its domain only if its energy drops under the threshold ω_{\min} . With that in mind, it is natural that each entry of $S^{(k)}(t)$ must monitor the current energy value of its corresponding particle k . If this energy ever drops under a given threshold, the switch must be enabled. Analogously, if the particle still has an energy value greater than this threshold, then the switch should be disabled. Mathematically, the k th-entry of $S(t)$ can be written as:

$$S^{(k)}(t) = \mathbf{1}_{\{E^{(k)}(t) = \omega_{\min}\}}, \quad (3.14)$$

where $\dim(S(t)) = 1 \times K$. Specifically, $S^{(k)}(t) = 1$ if $E^{(k)}(t) = \omega_{\min}$ and 0, otherwise. The upper limit, ω_{\max} , has been introduced to prevent any particle in the network from increasing its energy to an undesirably high value and, therefore, taking a long time to become exhausted even if it constantly visits vertices from rival particles. In this way, the community and cluster detection rates of the proposed technique would be considerably reduced.

In the following, we apply the remaining key concepts introduced so far in a concise and simple example.

Example 2. Consider the network depicted in Fig. 3.3. Suppose there are two particles, namely, red (dark gray) and blue (gray), each of which located at vertices 17 and 1, respectively. As both

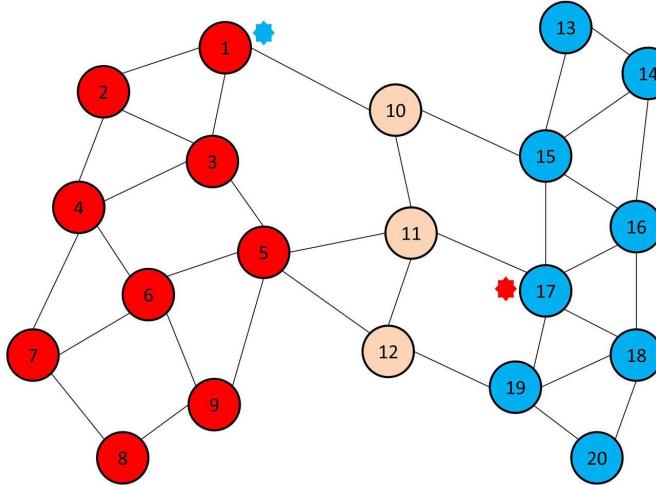


Figure 3.3: Illustration of the reanimation procedure in a typical situation. There are two particles, namely red (dark gray) and blue (gray), located at vertices 17 and 1, respectively, at time t . The network encompasses 20 vertices. As both particles are visiting vertices whose owners are rival particles, their energy will drop. In this example, suppose both energy levels of the particles reach the minimum possible value, ω_{\min} . The vertex color represents the particle that is imposing the highest domination level at time t . The beige (light gray) denotes a non-dominated vertex.

particles are visiting vertices whose owners are rival particles, their energy will drop. Consider, in this case, that both particles have reached the minimum allowed energy, i.e., ω_{\min} , at time t . Therefore, according to (3.14), both particles are exhausted. Consequently, $S^{(\text{red})}(t) = 1$ and $S^{(\text{blue})}(t) = 1$, and the transition matrix associated to each particle will only have the second term in the convex combination of (3.1). According to the mechanism of the dynamical system, these particles will be teleported back to their owned territory, in order to have their energy properly recharged, no matter if there is a physical connection or not between the vertices. This teleportation will take place on account of (3.12). In view of this scenario, the following transition matrix holds for the red (dark gray) particle at time t :

$$\mathbb{P}_{\text{transition}}^{(\text{red})}(i, j, t) = \frac{1}{9}, \forall i \in \mathcal{V}, j \in \{v_1, v_2, \dots, v_9\}, \quad (3.15)$$

$$\mathbb{P}_{\text{transition}}^{(\text{red})}(i, j, t) = 0, \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{v_1, v_2, \dots, v_9\}, \quad (3.16)$$

and the transition matrix associated to the blue (gray) particle at time t is written as:

$$\mathbb{P}_{\text{transition}}^{(\text{blue})}(i, j, t) = \frac{1}{8}, \forall i \in \mathcal{V}, j \in \{v_{13}, v_{14}, \dots, v_{20}\}, \quad (3.17)$$

$$\mathbb{P}_{\text{transition}}^{(\text{blue})}(i, j, t) = 0, \forall i \in \mathcal{V}, j \in \mathcal{V} \setminus \{v_{13}, v_{14}, \dots, v_{20}\}. \quad (3.18)$$

One can verify that, given that the particle is exhausted, it does not matter where the particle

is at the present moment, it will teleport back to its territory (set of dominated vertices). The determination of which of the dominated vertices to visit follows a uniform distribution, i.e., each owned vertex has an equal chance to be visited by the exhausted particle. This confinement mechanism prevents aimless navigation of the particle in the network, as it will be confirmed by the computer simulations in the following sections.

Once defined the collection of matrices associated to each particle, we couple all of them into a single representative transition matrix denominated $\mathbb{P}_{\text{transition}}(t)$ which models the transition of $p(t)$ to $p(t + 1)$. Given the current system's state at time t , one can see that $p^{(k)}(t + 1)$ and $p^{(u)}(t + 1)$ are independent for every pair $(k, u) \in \mathcal{K} \times \mathcal{K}, k \neq u$. Another way of looking at this fact is that, given the immediate past position of each particle, it is clear that, via (3.1), the next particle's location is only dependent on the topology of the network (random term) and the domination levels of the neighborhood (preferential term). In this way, $\mathbb{P}_{\text{transition}}(t)$ can be written as:

$$\mathbb{P}_{\text{transition}}(t) = \mathbb{P}_{\text{transition}}^{(1)}(t) \otimes \dots \otimes \mathbb{P}_{\text{transition}}^{(K)}(t), \quad (3.19)$$

where \otimes denotes the Kronecker tensor product operator. In this way, (3.19) completely specifies the transition distribution matrix for *all* the particles in the network.

Essentially, when $K \geq 2$, $p(t)$ will be a vector and we would no longer be able to conventionally define the row $p(t)$ of matrix $\mathbb{P}_{\text{transition}}(t)$. Owing to this, we define an invertible mapping $f : \mathcal{V}^K \mapsto \mathbb{N}$. The function f simply maps the input vector to a scalar number that reflects the natural ordering of the tuples in the input vector. For example, $p(t) = [1, 1, \dots, 1, 1]$ (all particles at vertex 1) denotes the first state; $p(t) = [1, 1, \dots, 1, 2]$ (all particles at vertex 1, except the last particle, which is at vertex 2) is the second state; and so on, up to the scalar state V^K . Therefore, with this tool, we can fully manipulate the matrix $\mathbb{P}_{\text{transition}}(t)$.

* * *

Remark 3. *The matrix $\mathbb{P}_{\text{transition}}(t)$ in (3.19) possesses dimensions $V^K \times V^K$, which are undesirably high. In order to save up space, one can use the individual transition matrices associated to each particle (therefore, we maintain a collection of K matrices), as shown in (3.1), each of which with dimensions $V \times V$, to model the particles' transition dynamics with no loss of generality, by using the following method: once every transition of the collection of K matrices has been performed, one could concatenate the new particle positions to assemble the stochastic vector that denotes the particles' localization, $p(t + 1)$, in an ordered manner. With this technique, the spatial complexity would not surpass $\mathcal{O}(KV)$, provided that we implement the matrices in a sparse mode.*

* * *

3.1.3 Defining the Stochastic Nonlinear Dynamical System

In light of the results obtained in the previous section, we are ready to enunciate the proposed dynamical system, which models the competition of particles in a given network. The internal state of the dynamical system is denoted as:

$$X(t) = \begin{bmatrix} p(t) \\ N(t) \\ E(t) \\ S(t) \end{bmatrix}, \quad (3.20)$$

and the proposed competitive dynamical system is given by:

$$\phi : \begin{cases} p^{(k)}(t+1) = j, & j \sim \mathbb{P}_{\text{transition}}^{(k)}(t) \\ N_i^{(k)}(t+1) = N_i^{(k)}(t) + \mathbb{1}_{\{p^{(k)}(t+1)=i\}} \\ E^{(k)}(t+1) = \begin{cases} \min(\omega_{\max}, E^{(k)}(t) + \Delta), & \text{if } \text{owner}(k, t) \\ \max(\omega_{\min}, E^{(k)}(t) - \Delta), & \text{if } \neg \text{owner}(k, t) \end{cases} \\ S^{(k)}(t+1) = \mathbb{1}_{\{E^{(k)}(t+1)=\omega_{\min}\}} \end{cases} \quad (3.21)$$

The first equation of system ϕ is responsible for moving each particle to a new vertex j , where j is determined according to the time-varying transition matrix in (3.1). In other words, the acquisition of $p(t+1)$ is performed by generating random numbers following the distribution of the transition matrix $\mathbb{P}_{\text{transition}}^{(k)}(t)$. The second equation updates the number of visits that vertex i has received by particle k up to time t ; the third equation is used to maintain the current energy levels of all the particles inserted in the network; and the fourth equation indicates whether the particle is active or exhausted, depending on its actual energy level. Note that system ϕ is nonlinear. This occurs on account of the indicator function, which is nonlinear.

Observe that system ϕ can also be written in matrix form as:

$$\phi' : \begin{cases} p(t+1) = f_p, & f_p \sim \mathbb{P}_{\text{transition}}(t) \\ N(t+1) = f_N(N(t), p(t+1)) \\ E(t+1) = f_E(N(t+1), p(t+1)) \\ S(t+1) = f_S(E(t+1)) \end{cases}, \quad (3.22)$$

where f_p , $f_N(\cdot)$, $f_E(\cdot)$, and $f_S(\cdot)$ are suitable stochastic matrix functions, whose entries

have been defined in (3.21). An important characteristic of system ϕ , which will be extensively used later, is its Markovian property (see Proposition 1).

3.1.4 Setting up the Initial Conditions of the Dynamical System

In order to run the system ϕ , initial conditions of the variables should be defined. First, the particles are randomly inserted into the network, i.e., the values of $p(0)$ are randomly set. The initial positions of the particles do not affect the community detection or data clustering results, because each of them will be confined into a different community or cluster due to the competitive nature of the technique. This will occur even if they are put together at the beginning.

Each entry of matrix $N(0)$ is initialized according to the following expression:

$$N_i^{(k)}(0) = \begin{cases} 2, & \text{if particle } k \text{ is generated at vertex } i \\ 1, & \text{otherwise} \end{cases}. \quad (3.23)$$

Since a fair competition amongst the particles is desired, we place isonomy in their initial energy values, i.e., all particles $k \in \mathcal{K}$ start out with the same energy level given by:

$$E^{(k)}(0) = \omega_{\min} + \left(\frac{\omega_{\max} - \omega_{\min}}{K} \right). \quad (3.24)$$

Lastly, all particles are set to the active state at the beginning of the process:

$$S^{(k)}(0) = 0. \quad (3.25)$$

3.1.5 Detailing the Algorithm

In order to facilitate the understanding of how the proposed stochastic dynamical system evolves in time, Fig. 3.4 depicts a flowchart with the main tasks that must be processed in the proposed technique. In the first block, “Set Initial Conditions,” we need to initialize the system state $X(0)$, which is composed of a joint distribution of $N(0)$, $p(0)$, $E(0)$, and $S(0)$. After that, the system runs and the “Stop Criterion” logical statement is checked at each iteration. For a specific iteration, each particle needs to move to a next vertex according to the transition matrix and its current position. This is exactly done by the inner loop starting from the “ $k < K$ ” logical statement. Inside this loop, we generate the time-variant transition matrix associated to particle k

(“Calculate Transition Matrix of Particle k ” block) and we move the particle to a next vertex according to this matrix (“Particle k Visits Another Vertex” block). When all particles have properly made their movements, the inner loop ceases and we update the remaining system variables, i.e., $N(t)$, $E(t)$, and $S(t)$, for some $t \geq 1$ (“Update Remaining System Variables” block). After that, we have completely assembled the system state $X(t)$ and we can either move forward to a next iteration or, instead, if the stop criterion has been met, supply $\bar{N}(t)$ to the user (“Return Domination Level Matrix” block).

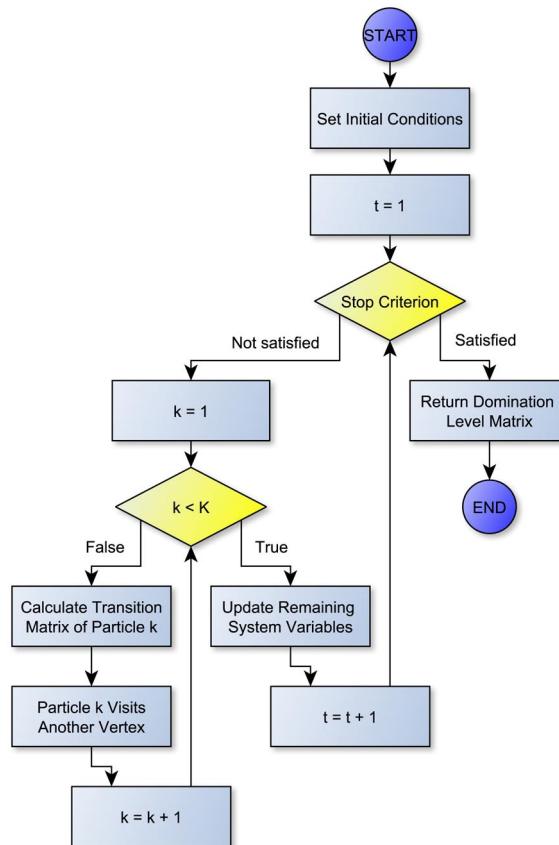


Figure 3.4: A flowchart of the methodology to iterate the stochastic dynamical system ϕ .

Algorithm 1 summarizes all the steps to iterate the system ϕ . Essentially, the algorithm accepts the data set (data) and four user-defined parameters: the number of particles (K), the desired fraction of preferential movement (λ), the energy that each particle gains or loses (Δ), and a stopping factor (ϵ). Usually, good results can be obtained by selecting $\lambda \in [0.2, 0.8]$ and $\Delta \in [0.1, 0.4]$. ϵ can be set to an arbitrary small value. In all simulations in this section, we simply use $\epsilon = 0.05$. The empirical determination of the aforementioned values is going to be investigated in the following sections. Observe that, in Step 22, we utilize the matricial max-norm $\|\cdot\|_\infty$ to test the termination criterion. Such operator yields the maximum absolute row sum of the matrix. Note that the stopping criterion can also be defined in terms of a maximum

number of allowed iterations. The only parameter that needs to be calibrated accordingly to each input data set is the number of particles K , whose determination will be described later.

Algorithm 1 The Particle Competition Algorithm.

```

1: procedure PARTICLECOMPETITION( $K$ , data,  $\lambda$ ,  $\Delta$ ,  $\epsilon$ )
2:    $A \leftarrow \text{BUILDGRAPH}(\text{data})$ 
3:    $p(0) \leftarrow \text{GENERATEPARTICLESATRANDOM}(A)$ 
4:    $P_{\text{rnd}} \leftarrow \text{CALCULATERANDOMMATRIX}(A)$ : Use (3.2)
5:    $N(0) \leftarrow \text{CALCULATEINITIALN}(p(0))$ : Use (3.23)
6:    $\bar{N}(0) \leftarrow \text{CALCULATENBAR}(N(0))$ : Use (3.7)
7:    $E(0) \leftarrow \text{CALCULATEINITIALE}(K)$ : Use (3.24)
8:    $S(0) \leftarrow \text{CALCULATEINITIALS}()$ : Use (3.25)
9:    $t \leftarrow 1$ 
10:  repeat
11:    for  $k = 1$  to  $K$  do
12:       $P_{\text{pref}}^{(k)}(t) \leftarrow \text{CALCULATEPPREF}(N(t-1), p(t-1))$ : Use (3.8)
13:       $P_{\text{rean}}^{(k)}(t) \leftarrow \text{CALCULATEPREAN}(N(t-1), p(t-1))$ : Use (3.12)
14:       $P_{\text{tran}}^{(k)}(t) \leftarrow \text{SETPTRAN}(\lambda, P_{\text{rnd}}, P_{\text{pref}}^{(k)}(t), P_{\text{rean}}^{(k)}(t))$ : Use (3.1)
15:       $p^{(k)}(t) \leftarrow \text{CHOOSENEXTVERTICES}(P_{\text{tran}}^{(k)}(t), p^{(k)}(t-1))$ 
16:    end for
17:     $N(t) \leftarrow \text{UPDATEN}(N(t-1), p(t))$ : Use 1st eq. in (3.21)
18:     $\bar{N}(t) \leftarrow \text{CALCULATENBAR}(N(t))$ : Use (3.7)
19:     $E(t) \leftarrow \text{UPDATEE}(\Delta, E(t-1), \bar{N}(t), p(t))$ : Use 2nd eq. in (3.21)
20:     $S(t) \leftarrow \text{UPDATES}(E(t))$ : Use 3rd eq. in (3.21)
21:     $t \leftarrow t + 1$ 
22:  until  $\|\bar{N}(t) - \bar{N}(t-1)\|_\infty < \epsilon$ 
23:  return  $\bar{N}(t)$ 
24: end procedure

```

3.1.6 Discovering the Computational Complexity

The whole algorithm for data clustering (Algorithm 1) can be divided into two parts: (i) Step 2: network construction from the input data set; (ii) Steps 3 to 23: definition of the community detection algorithm. The following list provides a computational complexity analysis of each step of the algorithm:

- *Step 2:* Construction of the graph from the input data set. It has complexity order $\mathcal{O}(V^2)$, since the distance matrix must be evaluated;
- *Step 3:* Generation of K particles in a random manner (uniform distribution). This has complexity order $\mathcal{O}(K)$;
- *Step 4:* In this step, we must visit every link of the network. Thus, its complexity order is $\mathcal{O}(E)$, where E denotes the number of links;
- *Steps 5 and 6:* An assignment is made for each of the $K \times V$ entries of $N(0)$ and $\bar{N}(0)$, respectively. Hence, these steps have complexity order $\mathcal{O}(KV)$;

- *Steps 7 and 8:* A simple assignment is performed for each of the K entries of $E(0)$ and $S(0)$. In this way, the complexity order is $\mathcal{O}(K)$;
- *Step 12:* Suppose that $\langle k \rangle$ is the average vertex degree of the network, it follows that this can be performed in $\mathcal{O}(\langle k \rangle)$;
- *Step 13:* We maintain a hashtable to store the vertices that are being owned by each particle. In this way, it takes constant time to find out a vertex dominated by an exhausted particle, i.e., $\mathcal{O}(1)$;
- *Step 14:* Scalar multiplication with each neighbor of the vertex that particle k is visiting. This is finished in $\mathcal{O}(\langle k \rangle)$ time;
- *Step 15:* Particle k chooses the next vertex to visit. We use a cumulative probability function, whose probability density function follows the distribution calculated in Step 14, and, by generating a random number following a uniform distribution $r \sim U(0, 1)$, we perform the transition to a next vertex. Therefore, as the transition matrix possesses arbitrary probability distribution, this is done in $\mathcal{O}(\langle k \rangle)$ time;
- *Steps 17 and 18:* Update matrices $N(t)$ and $\bar{N}(t)$. Considering that at most K different vertices will be visited at any given time, then it is guaranteed that at most K rows of the matrices $N(t)$ and $\bar{N}(t)$ will be changed. So, this update can be done in $\mathcal{O}(K^2)$ time, on account that each of the K rows has K entries;
- *Steps 19 and 20:* Completed in $\mathcal{O}(K)$ time.

Since Steps 12 to 15 repeat K times, it follows that this block has complexity order $\mathcal{O}(K\langle k \rangle)$. The complexity order of the next block, defined by Steps 17 to 21, is determined by Step 17 or 18, i.e., $\mathcal{O}(K^2)$. Therefore, the community detection algorithm without the “repeat” loop has complexity order $\mathcal{O}(K\langle k \rangle + K^2)$.

Now we estimate the number of iterations of the “repeat” loop. Consider a network with some completely separated communities and suppose that each community has a single particle. The ownership of each vertex can be determined by a single visit of the particle; thus, the number of iterations of the main loop is certainly $\mathcal{O}(V) = c_1 V$, where c_1 is a positive constant proportional to the fraction of random movement performed by the particles. If the communities are connected in a well-defined manner (there are few intercommunity connections), the ownership of each vertex can be determined by a small number of visits. Then, in order to have all V vertices dominated by the particles, the number of iterations is again $\mathcal{O}(V) = c_2 V$, where c_2 is a positive constant satisfying $c_2 > c_1$. Following the same reasoning, we conclude that the number of iterations required for all vertices to be completely dominated by the particles is $\mathcal{O}(V) = cV$, where c is a constant whose magnitude increases with the portion of

intercommunity links. Therefore, we have estimated that the main loop repeats cV times.

In summary, the community detection algorithm (Algorithm 1 without Step 2) has complexity order $\mathcal{O}(K\langle k \rangle V + K^2V)$. Some specific cases can be observed:

- If the network is sparse, i.e., $\langle k \rangle \ll V$, the community detection part runs in $\mathcal{O}(K^2V)$ time;
- If the average degree $\langle k \rangle$ is proportional to V (a highly connected network), the community detection part runs in $\mathcal{O}(KV^2)$ time;
- Since the quantity of particles inserted in the network is usually small and many real-world networks are sparse, i.e., $K \ll V$ and $\langle k \rangle \ll V$, it is reasonable to assume that the community detection algorithm has linear complexity order ($\mathcal{O}(V)$) in the majority of the cases.

For the data clustering problem, we must transform the vector-based data set into a graph. This task is done by Step 2: building the adjacency matrix from the input data set, which takes $\mathcal{O}(V^2)$ time. Thus, the computational complexity of the whole algorithm is $\mathcal{O}(V^2 + K\langle k \rangle V + K^2V)$. Two particular situations can be pointed out:

- For sparse networks, the complexity of the proposed algorithm for data clustering is dominated by Step 2. In this case, the complexity order is $\mathcal{O}(V^2)$;
- If the network is dense, then the computational complexity of the proposed technique is dominated by the main loop of the algorithm. In this way, the complexity order becomes $\mathcal{O}(KV^2)$.

For the sake of completeness, we compare several representative community detection techniques with the proposed technique in Table 3.1¹. For an extensive list, see Table 1 in (Danon *et al.*, 2005) and Table 1 in (Fortunato, 2010).

Regarding the data clustering techniques, Table 3.2 outlines the computational complexity of our algorithm, along with some representative techniques developed so far.

Based on the analysis and comparisons of Tables 3.1 and 3.2, we conclude that the proposed community detection technique has the lowest computational complexity order compared to all other techniques developed so far. With respect to data clustering

¹It is worth noting that the community detection technique with the lowest order of computational complexity developed so far is the Voltage Drop Approach (Wu and Huberman, 2004), which has time complexity $\mathcal{O}(V + L)$, i.e., it is linear $\mathcal{O}(V)$ for sparse networks and $\mathcal{O}(V^2)$ for non-sparse networks. Thus, this technique has the same complexity order of our method. We have not put the Voltage Drop Approach in Table 3.1 because it is not a representative community detection technique, due to the following drawbacks: (i) the number of communities must be known *a priori*; and (ii) the technique needs to know at least one vertex, whose community label is known, for each community of the graph.

tasks, the proposed technique has a reasonable complexity order (it is not the lowest one), mainly by virtue of the network formation step.

Table 3.1: Time complexity of community detection techniques.

Technique	Time Complexity	Ref.
Betweenness	$\mathcal{O}(V^2E)$	(Girvan and Newman, 2002)
Extremal Optimization	$\mathcal{O}(V^2 \log(V))$	(Duch and Arenas, 2005)
Modularity	$\mathcal{O}(V \log^2(V))$ - Sparse Network, $\mathcal{O}((V+E)V)$ - Non-sparse Network	(Newman and Girvan, 2004) (Clauset <i>et al.</i> , 2004)
Spectral Approach	$\mathcal{O}(V^2)$	(Capocci <i>et al.</i> , 2004)
Overlapping Community	$\mathcal{O}(\exp(V))$	(Palla <i>et al.</i> , 2005)
Proposed Technique	$\mathcal{O}(K^2V)$ - Sparse Network, $\mathcal{O}(KV^2)$ - Non-sparse Network	-

3.1.7 Method for Determining the Optimal Number of Particles

In this section, we present a method for determining the optimal number of particles to be inserted into the network. In order to do so, an evaluator index that monitors the information generated by the competitive model itself is employed. To this end, the average maximum domination level $\langle R(t) \rangle \in [0, 1]$ will be utilized, which is given by the following expression:

$$\langle R(t) \rangle = \frac{1}{V} \sum_{u=1}^V \max_{m \in \mathcal{K}} \left(\bar{N}_u^{(m)}(t) \right), \quad (3.26)$$

where $\bar{N}_u^{(m)}(t)$ indicates the domination level that particle m is imposing on vertex u at time t (see (3.7)) and $\max_{m \in \mathcal{K}} \left(\bar{N}_u^{(m)}(t) \right)$ yields the maximum domination level imposed on vertex u at time t . For a given network which presents some communities, say K communities, if we add exactly K particles, each of them will dominate a community; thus, a particle will not interfere much in the acting region of the other particles. As a consequence, $\langle R(t) \rangle$ will be large. In the extreme case, if each vertex is completely dominated by a single particle, $\langle R(t) \rangle$ reaches 1. However, if we add more than K particles, inevitably it will occur that more than one particle shares the same community. In this case, they will dispute the same group of vertices. By virtue of that, one will lower the domination levels imposed by the other particles, and vice versa. As a result,

² V is the number of data items, K is the number of clusters, M is the data dimension, and I is the number of iterations.

³The Modularity Greedy Algorithm, originally proposed only for community detection tasks, has been, here, adapted for data clustering tasks. The first step is to construct a network using the k -NN approach. After that, we apply the algorithm described in (Clauset *et al.*, 2004).

Table 3.2: Time complexity of data clustering techniques.

Technique	Time Complexity	Ref.
CHAMELEON	$\mathcal{O}(V^2)$	(Theodoridis and Koutroumbas, 2008)
Expectation Maximization ²	$\mathcal{O}(K^2 IVM)$	(Dempster <i>et al.</i> , 1977)
Fuzzy C-Means ²	$\mathcal{O}(IVM)$	(Bezdek, 1981)
Optimized K-Means ²	$\mathcal{O}(IVM)$	(Khan, 2004)
Modularity ³	$\mathcal{O}(V^2)$ - Sparse Network, $\mathcal{O}((V + E)V)$ - Non-sparse Network	(Clauset <i>et al.</i> , 2004) (Clauset <i>et al.</i> , 2004)
Particle Swarm	$\mathcal{O}(V^2)$	(Shanmugam <i>et al.</i> , 2010)
Proposed Technique	$\mathcal{O}(V^2)$ - Sparse Network, $\mathcal{O}(KV^2)$ - Non-sparse Network	-

$\langle R(t) \rangle$ will be low. Conversely, if we insert in the network a quantity of particles less than the number of clusters (K), a strong competition will also occur amongst the particles. In this case, they will attempt to dominate more than one community at once. Again, $\langle R(t) \rangle$ will be low. Therefore, this scenario suggests that the actual number of clusters can be estimated by checking the maximum value of $\langle R(t) \rangle$.

If we apply this evaluator index to assess how many communities or clusters a data set possess, we have to independently run the stochastic dynamical system several times. In practical terms, the number of particles must vary from 2 until K' particles, where K' is a slightly larger constant than K , which here denotes the actual number of communities or clusters in the data set. In this way, for community detection tasks, the complexity order of the proposed algorithm would approximately become $\mathcal{O}(K^3 V)$ for sparse networks and $\mathcal{O}(K^2 V^2)$ for densely connected networks. Regarding data clustering tasks, it would approximately become $\mathcal{O}(KV^2)$ for sparse networks and $\mathcal{O}(K^2 V^2)$ for dense networks. Having in mind that the number of clusters is small, i.e., it is far less than the number of data items ($K \ll V$), this checking process may not change the complexity order of the proposed community detection and data clustering algorithms.

We may now proceed to empirically test our measure in random clustered networks with different numbers of communities. In Fig. 3.5a, we show the community detection rate Ψ reached by our algorithm for networks presenting 5, 10, and 15 communities, respectively, whereas Fig. 3.5b reveals the corresponding $\langle R(t) \rangle$ achieved each of the three corresponding scenarios. One can see that the maximum $\langle R(t) \rangle$ occurs exactly when the number of particles is equal to the number of communities in the network. Moreover, we can empirically verify that after this maximum has been reached, subsequent insertions of more particles will only decrease the community detection rate and the average maximum domination level.

In order to further verify our method, Fig. 3.6 shows the application of this evaluator index on real-world data sets (UCI), namely Iris (3 classes), Credit Approval (2

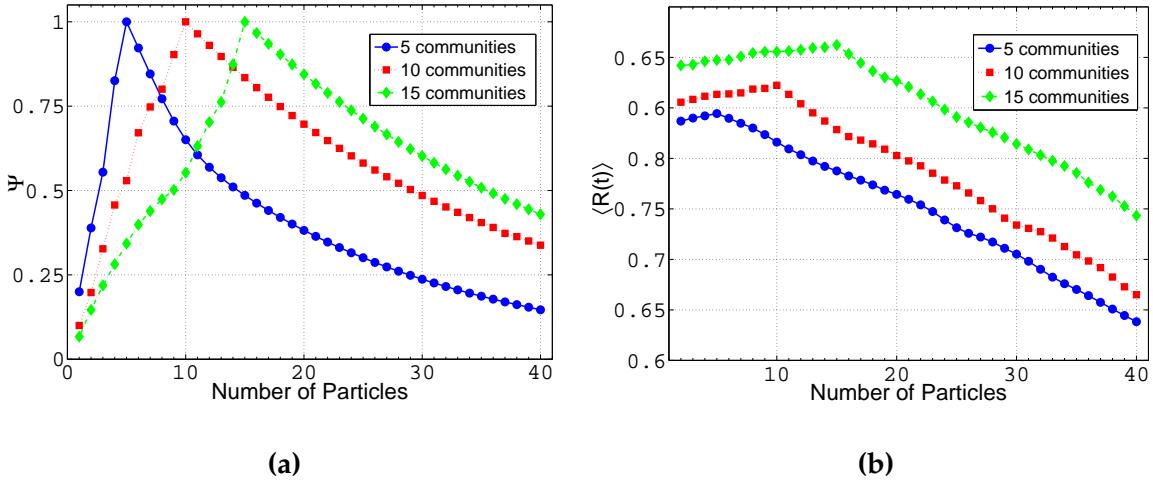


Figure 3.5: Determination of the optimal number of particles K in random clustered networks. (a) Correct community detection rate reached by the algorithm vs. K . Networks with 5 (160 vertices), 10 (320 vertices), and 15 (480 vertices) communities are used. (b) $\langle R(t) \rangle$ vs. K . The corresponding $\langle R(t) \rangle$ for the networks in (a). The constructed networks have the following characteristics: $\langle k \rangle = 15$, and $z_{\text{out}}/\langle k \rangle = 0.3$. Results are averaged over 20 simulations.

classes), Vowel (11 classes), and Letter Recognition (26 classes) data sets. One can verify that the maximum $\langle R(t) \rangle$ happens precisely when the number of particles exactly matches the number of classes (clusters) in the problem. This confirms that $\langle R(t) \rangle$ is a good measure for determining the optimal number of particles as well as estimating the actual number of clusters of a data set.

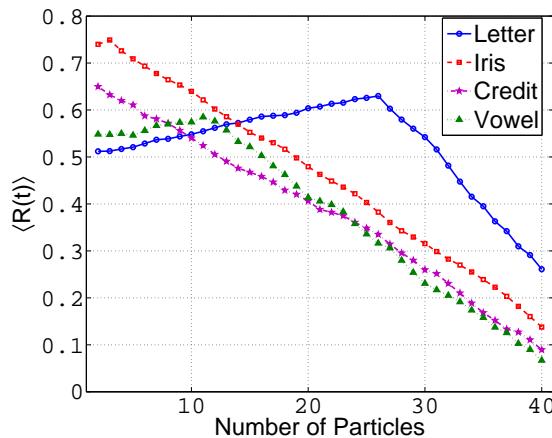


Figure 3.6: Determination of the optimal number of particles K (the actual number of clusters) in real-world data sets. In all these simulations, we have used the k -nearest neighbor network formation technique with $k = 4$. The number of classes that each data set originally possesses is: (i) Iris: 3, (ii) Credit Approval: 2, (iii) Vowel: 11, and (iv) Letter Recognition: 26. Results are averaged over 20 simulations.

3.1.8 Method for Detecting Overlapping Structures

The competitive process carries a rich set of information throughout time. With the aid of such information, we are going to derive a measure that detects overlapping structures or vertices in a given network. For this matter, it is worth noticing that the domination level matrix $\bar{N}(t)$ can be used to indicate whose vertices are members of just one or several groups or communities in the following way: if the maximum domination level imposed by an arbitrary particle k on a specific vertex i is much greater than the second maximum domination level imposed by another particle on the same vertex, then we can conclude that this vertex is being strongly dominated by particle k and no other particle is influencing it in a relevant manner. Therefore, the overlapping nature of such vertex is minimal. On the other hand, when these two quantities are similar, then we can infer that the vertex in question holds an inherently overlapping characteristic. In light of these considerations, we can mathematically model this behavior as follows: let $M_i(x, t)$ denote the x th greatest domination level value imposed on vertex i at time t . In this way, the overlapping index of vertex i , $O_i(t) \in [0, 1]$, is given by:

$$O_i(t) = 1 - (M_i(1, t) - M_i(2, t)), \quad (3.27)$$

i.e., the overlapping index $O_i(t)$ measures the gap between the two greatest domination levels imposed by any pair of particles in the network on vertex i . Succinctly, when this gap is high, a strong domination is taking place on that vertex and, hence, $O_i(t)$ yields a low value. On the other hand, when the competition is fiercely occurring on vertex i , some domination levels imposed on that vertex are expected to reside, somewhat, near each other. Consequently, the gap between the two greatest domination levels is hoped for being low, producing a large value for the overlapping index $O_i(t)$.

Now, we take a tour to assess the impact of the routine that detects overlapping vertices in the particle competition algorithm. Equation (3.27) is only applied in the final iteration of the dynamical process in an independent manner. Analyzing this equation, we see that a linear scan over the vertices is necessary. Furthermore, the calculations that are performed on a vertex are simple: we need to find the 2 largest domination levels imposed on it. We can encounter this in $\mathcal{O}(K)$ time, since there are K entries for each vertex. In view of this, the time complexity of only the overlapping index is $\mathcal{O}(KV)$.

In summary, the task of detecting overlapping vertices in communities becomes $\mathcal{O}(K^2V) + \mathcal{O}(KV) = \mathcal{O}(K^2V)$ and in data clustering becomes $\mathcal{O}(KV^2) + \mathcal{O}(KV) = \mathcal{O}(KV^2)$. In this way, the detection of overlapping vertices does not increase the time complexity of the model, which is very interesting in real-world applications.

For the sake of completeness, we compare representative overlapping cluster detection techniques with our proposed technique in Table 3.3². We can infer from this table that the proposed overlapping cluster detection technique has the lowest complexity order compared to all other techniques developed so far.

Table 3.3: Time complexity of well-known overlapping cluster detection techniques.

Technique	Time Complexity	Ref.
Fuzzy C-Means Clustering ⁴	$O(E\phi h + V\phi^2 h + \phi^3 h)$	(Zhang <i>et al.</i> , 2007)
EAGLE ⁵	$O(V^2 s)$ - Sparse Network $O(\exp(V))$ - Non-sparse Network	(Shen <i>et al.</i> , 2009)
Extended Modularity ⁶	$O(CV^2)$	(Nicosia <i>et al.</i> , 2009)
Fitness-Function Optimization	$O(V^2)$	(Lancichinetti <i>et al.</i> , 2009)
Overlapping Community	$O(\exp(V))$	(Palla <i>et al.</i> , 2005)
Proposed Technique	$O(K^2 V)$ - Sparse Network $O(KV^2)$ - Non-sparse Network	-

3.2 Empirical Analysis of the Technique

In this section, we aim at providing guidelines for using the proposed technique and we also dive into the problem of whether the proposed technique really converges. This material serves as a gist and as a motivating tool to use the particle competition method.

3.2.1 Parameter Sensitivity Analysis

The proposed competitive model requires a set of parameters to work. In this section, we will show the impact of each one of them, except for the K (number of particles) and the ϵ (termination criterion) parameters. The parameter K , which determines the number of particles put into the network, will be derived using the heuristic for estimating the actual number of clusters or communities presented in Section 3.1.7. The left-off parameter, i.e., ϵ , will receive a special attention in the following section, where we will discuss, by means of a convergence analysis, several natural approaches for terminating the algorithm. Moreover, we will reveal the reason behind why we have selected the matricial max-norm criteria for terminating the algorithm in detriment to

⁴ h is the number of iterations until the system converges. Essentially, the complexity of the algorithm is dominated by the calculation of the ϕ greatest eigenvalues.

⁵ s is the number of join operations to be performed. In brief, the step responsible for defining the cliques is the most time-consuming one.

⁶ C is the number of communities in the network. The critical point of this algorithm is the evaluation of the Q-fitness function.

the other approaches. Following the conclusion of this mathematical reasoning, we supply guidelines for the termination criteria ϵ .

Impact of the λ Parameter

The parameter λ is responsible for counterweighting the proportion of preferential and random walks performed by all particles in the network. Here, we study the parameter λ by making use of the networks generated by the method proposed in (Lancichinetti *et al.*, 2008), which has the following interesting characteristics: (i) the vertices of the network may possess different degrees and (ii) the communities are, in general, unbalanced. This was devised because real-world networks are characterized by heterogeneous distributions of vertex degree, whose tails often decay as power laws. The constructed networks assume that both degree and community size distributions follow a power law function, with exponents γ and β , respectively. Typical values of real-world networks are: $2 \leq \gamma \leq 3$ and $1 \leq \beta \leq 2$. Moreover, a mixing parameter μ is employed to interconnect communities in the following manner: each vertex shares a fraction $1 - \mu$ of its links with the other vertices of the same community and a fraction μ with the vertices of other communities. The generated networks have $V = 10\,000$ vertices and $\langle k \rangle = 15$. The benchmark process consists in varying the mixing parameter μ and in evaluating the resulting accuracy.

Figures 3.7a to 3.7d display how the cluster detection accuracy of the model behaves as we vary λ from 0 (pure random walks) to 1 (pure preferential walks) in the networks constructed using the methodology described in (Lancichinetti *et al.*, 2008) with different values of γ and β . As one can verify from the figure, this parameter is sensible to the outcome of the technique. Oftentimes, the optimal cluster detection accuracy is reached when a mixture of random and preferential walks occurs. Using a conservative approach, for $0.2 \leq \lambda \leq 0.8$, the model gives good accuracy results when applied to networks with communities.

Impact of the Δ Parameter

The parameter Δ is responsible for updating the energies of the particles, according to which type of vertex they are visiting. We will apply the same networks utilized in the previous analysis. Figures 3.8a to 3.8d portray the cluster detection accuracy achieved by the algorithm for distinct values of Δ . We can conclude that, for intermediate values of Δ , namely $0.1 \leq \Delta \leq 0.4$, the model is not sensitive to Δ . In contrast to that, as Δ gets higher and higher, the performance of the technique is reduced. This happens because the particles get exhausted as soon as they visit a vertex from the rival particles, because, in light of (3.13), the minimum energy ω_{\min} tends to be reached faster. In this way, it is inviable for the particles to switch the ownership of already

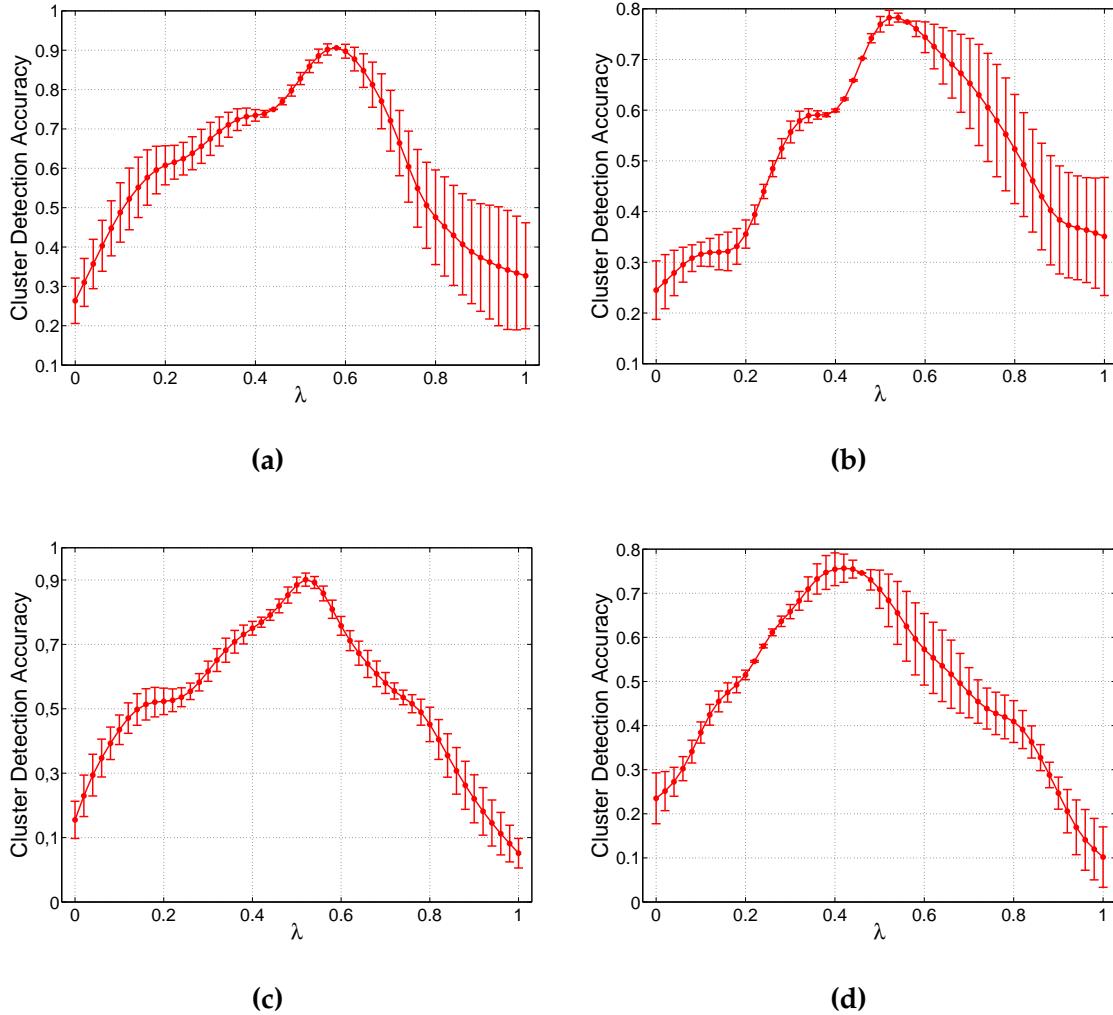


Figure 3.7: Cluster detection accuracy vs. parameter λ . We fix $\Delta = 0.15$. Taking into account the steep peak that is verified and the large negative derivatives that surround it, one can see that the parameter λ is sensible to the overall model's performance. Results are averaged over 30 simulations. (a) $\gamma = 2$ and $\beta = 1$; (b) $\gamma = 2$ and $\beta = 2$; (c) $\gamma = 3$ and $\beta = 1$; and (d) $\gamma = 3$ and $\beta = 2$.

conquered vertices. We can conceive this phenomenon as an artificial “hard labeling.” On the other hand, for too small Δ values, the particles are free to travel around the network with no energy penalties, i.e., they will rarely get exhausted. Therefore, all vertices in the network will be in constant competition and no class borders will be established and consolidated. In sum, Δ is not a very sensitive parameter for the model's performance, because the interval of Δ that yields reasonable results is significantly large.

Impact of the ω_{\min} and the ω_{\max} Parameters

The parameters ω_{\min} and ω_{\max} are used to limit the minimum and maximum energy levels of the particles in the network. These parameters need not to be analyzed,

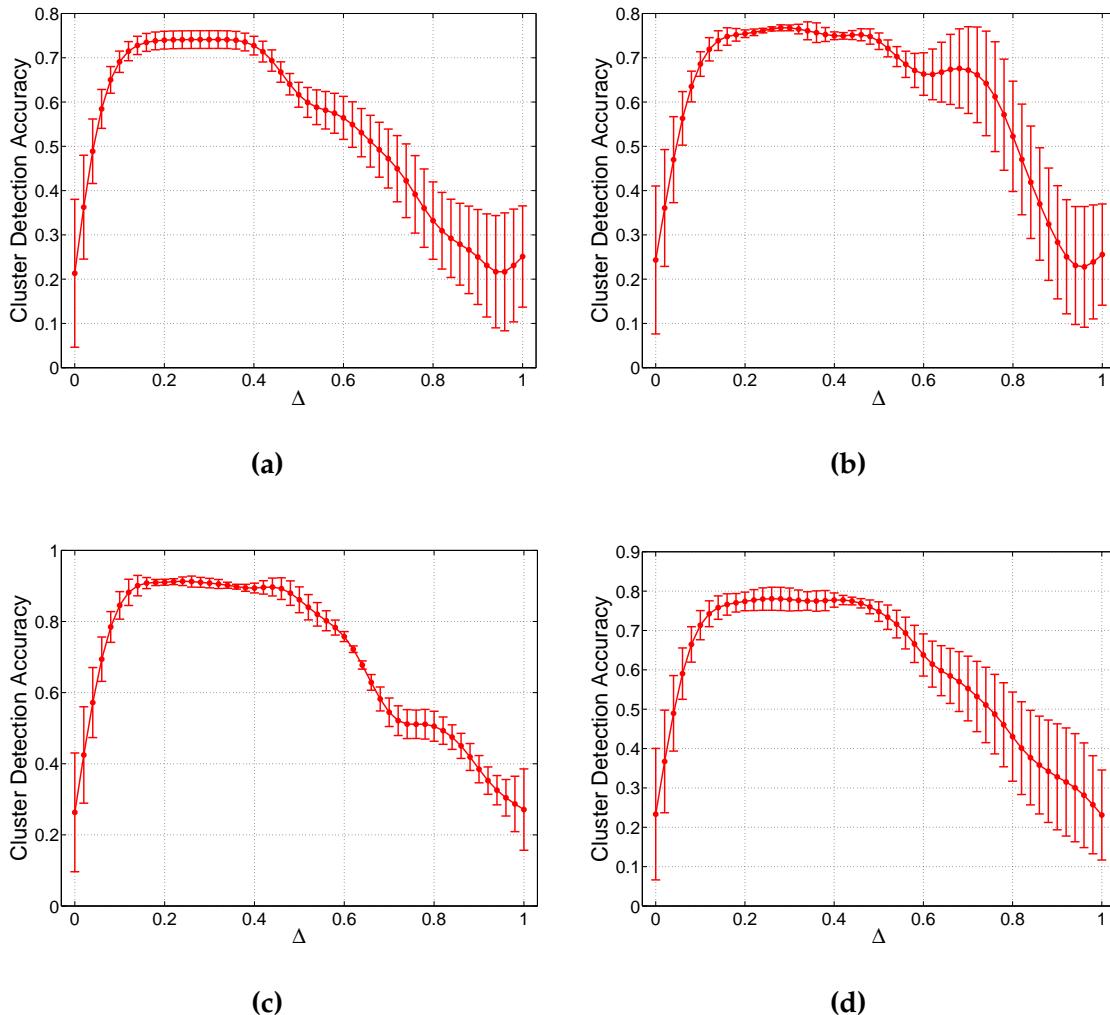


Figure 3.8: Cluster detection accuracy vs. parameter Δ . We fix $\lambda = 0.6$. Taking into account the large steady region that is verified, one can see that the parameter Δ is not sensible to the overall model's performance if one correctly uses it. Results are averaged over 30 simulations. (a) $\gamma = 2$ and $\beta = 1$; (b) $\gamma = 2$ and $\beta = 2$; (c) $\gamma = 3$ and $\beta = 1$; and (d) $\gamma = 3$ and $\beta = 2$.

since they are simply defining an interval and can be derived using Δ . Say that we have an interval $[0 - 1]$ and another $[0 - 10]$. We could use a Δ ten times larger in the second case and we would have the same behavior for the algorithm. In view of that, we standardize these by simply using $\omega_{\min} = 0$ and $\omega_{\max} = 1$.

Impact of the K Parameter

The parameter K , which quantifies the number of particles in the network, is the most sensitive parameter for the model's performance. Hence, its correct determination, or at least estimation, must be made with care. The heuristic presented for estimating the actual number of clusters or communities in the prior sections is a perfect candidate for deriving a proper value for the K parameter. In order to minimize the

error, we choose a K , named $K_{\text{candidate}}$, such as to maximize the the measure $\langle R(t) \rangle$, $\langle R(t)_{\max} \rangle$, as follows:

$$K_{\text{candidate}} \in \mathbb{N} : \langle R(t) \rangle = \langle R_{\max}(t) \rangle \quad (3.28)$$

$$= \arg \max_{K \in \mathbb{N}} \frac{1}{V} \sum_{u=1}^V \max_{m \in \mathcal{K}} (\bar{N}_u^{(m)}(t)) \quad (3.29)$$

$$\propto \arg \max_{K \in \mathbb{N}} \sum_{u=1}^V \max_{m \in \mathcal{K}} (\bar{N}_u^{(m)}(t)). \quad (3.30)$$

The reasoning behind this strategy is that, as we expect a dynamics equilibrium to be reached when each particle owns a specific cluster, it is of our best interest to choose the number of particles to be exactly equal to the number of clusters or communities in a data set.

Concluding Remarks and Guidelines Pervading the Parameters Selection

We have seen that Δ is not very sensitive to the model's performance, since the cluster detection accuracy remains steady for a large interval of Δ . We have also shown that ω_{\min} and ω_{\max} may not be considered primary parameters, because they are directly influenced by Δ . Finally, we have shown that the algorithm is somewhat sensitive to λ . Moreover, we have checked that K is very sensitive to the model's performance. Therefore, there are two important parameters to be calibrated, even though the λ parameter's influence is far less than K 's. Usually, λ can be set to lie within $[0.2, 0.8]$ (mixture of random and preferential walk) and K can be fairly well estimated using the provided heuristic.

Bearing in mind all this examination, unless specified otherwise, we always fix $\Delta = 0.1$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. Moreover, the ϵ parameter that is related to the termination criterion of the algorithm will also be investigated in the following sections. As we will see there, $\epsilon \leq 0.05$ provides reasonable results for the technique.

3.2.2 Convergence Analysis

One can wonder if the proposed particle competition model really converges, since the particles are always wandering around the vertices of the network in accordance with the combined random and preferential movement rules. This is a pertinent question and, in this section, we conduct an empirical analysis of the convergence of the model in an attempt to better understand its dynamics in the long run. One of our main concerns here is to elaborate on how we have derived the termination criterion used in Line 20 of Algorithm 1. For this purpose, we will investigate two natural ways

of terminating the proposed algorithm and we will show that the most suitable strategy is the one presented in Algorithm 1.

As an illustrative and meaningful example, consider the synthetic data sets shown in Figs. 3.9a to 3.9c. All of them are composed of two groups: the red or “circle” and the green or “square” groups. One can note that Fig. 3.9a shows a well-posed artificial data set, since the groups can be clearly distinguished in the geometrical space. Figure 3.9b displays a data set with somewhat well-posed groups, since a small region of each group overlaps with each other. Now, Fig. 3.9c depicts a rather difficult problem where the groups are not well-posed, because the overlapping region is constituted by a large portion of each group. In the latter, the clustering task is extremely difficult since the smoothness and cluster assumptions do not hold. This is reflected on the performance of the algorithms, which cannot, in their great majority, perform well in these kinds of data sets, since they basically rely on these hypotheses.

Now, we will investigate how the $\langle R(t) \rangle$ measure, which was previously introduced in Section 3.1.7, behaves as the proposed competitive dynamical system progresses in time. The simulation results with regard to the synthetic data sets given in Figs. 3.9a to 3.9c are depicted in Figs. 3.10a to 3.10c, respectively. In all these plots, we have explicitly indicated two important characteristics: (i) t_s , which is the time to reach the “almost-stationary” state of the model and (ii) the diameter of the region in which the almost-stationary state is confined within. Note that, since the competition is always taking place, the model will never reach a perfect stationary state. Rather, it will float around a quasi-stationary state because of the constant visits that the particles perform on the vertices of the network. These fluctuations are expected, since the random walk behavior of the particles, which is denoted by the second term in (3.1), compels the particles to venture through vertices that are not dominated by themselves. Now, if one chooses walks with no random behavior, i.e. only with preferential movements, these fluctuations would be eliminated, since the adventurous behavior of the particles would cease to exist. In this case, only the defensive behavior would be used by particles. As we will verify in the following sections, the model does not get good results when $\lambda = 1$ (only preferential walks), since the particles are obliged to keep visiting already dominated vertices, instead of broadening its region of dominance by visiting new non-dominated vertices.

One can see by Figs. 3.10a to 3.10c that the time to reach the almost-stationary state t_s lingers to be established as the overlapping region of the groups gets larger. From Fig. 3.10a to 3.10c, we have that t_s is given by 150, 430, 650, respectively. An argument that explain this phenomenon is that, since the groups’ borders become difficult to be properly distinguished, the competition among the particles in this overlapping region will fiercely be taking place across time. As a consequence, the regions of dominance of each particle will take longer to be consolidated and established. Another interesting

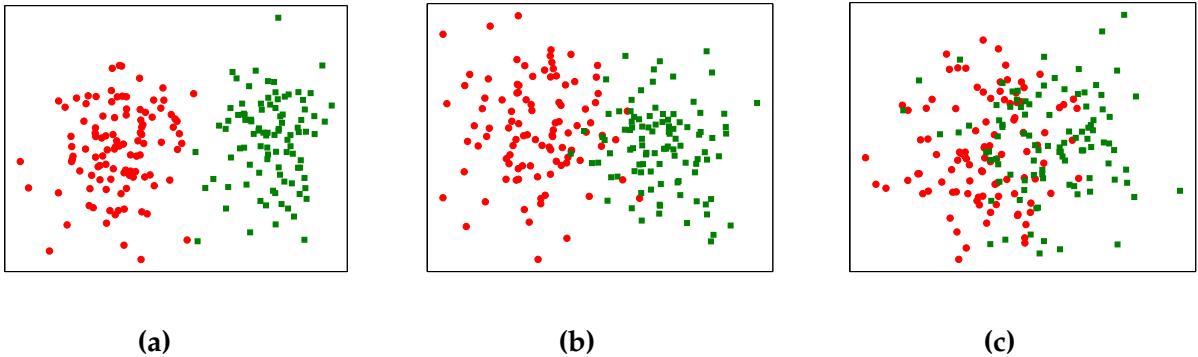


Figure 3.9: Scatter plot of artificial databases constituted by two groups. The data was derived from two bidimensional Gaussian distributions with varying mean and unitary covariance. (a) Well-posed groups; (b) Somewhat well-posed groups; (c) Not well-posed groups.

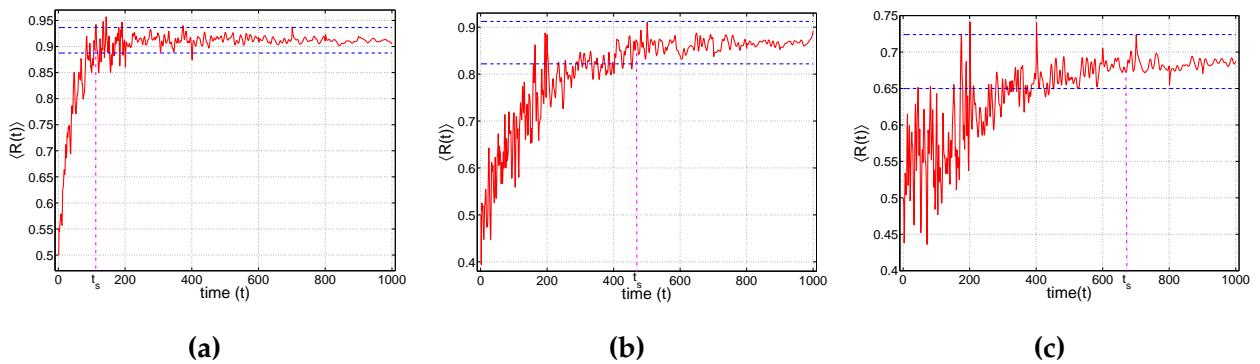


Figure 3.10: Convergence analysis of the particle competition algorithm when $\langle R(t) \rangle$ is used. The algorithm is run against the binary artificial databases, as one can see in Fig. 3.9. Here, we inspect how the $\langle R(t) \rangle$ measure varies as time progresses. Results obtained on (a) well-posed groups (Fig. 3.9a); (b) somewhat well-posed groups (Fig. 3.9b); (c) not well-posed groups (Fig. 3.9c).

phenomenon is that the diameter of the confinement region of $\langle R(t) \rangle$ grows larger as the mixture of the groups gets heavier. By Figs. 3.10a to 3.10c, the diameters of such regions are given by 0.06, 0.07, 0.08. This is expected by the same reasons we have stated before: the overlapping region is hard to be conquered by the particles since it is being constantly visited by rival particles. Therefore, $\langle R(t) \rangle$ will vary in this region, because the particle dominating each vertex will probably be changing before t_s is reached.

Next, we assess the behavior of $|\bar{N}(t+1) - \bar{N}(t)|_\infty$, which is the second candidate for our termination criterion, for the same synthetic data sets displayed in Figs. 3.9a to 3.9c. Figures 3.11a to 3.11c provide the corresponding simulation results. Interestingly, in the three cases under analysis, the variation of the $\bar{N}(t+1)$ in relation to $\bar{N}(t)$ (taking the max-norm) seems to reduce as time evolves. This happens because, as time progresses, the total number of visits performed by all particles always increases, since

each particle must visit at least a vertex in any given time. Now, looking at (3.7), which provides how the dominance level is calculated, we can see that the denominator always increases faster than the numerator; therefore, it provides an upper limit for $\bar{N}(t)$. In view of this, the variations from one iteration to another, i.e., $|\bar{N}(t+1) - \bar{N}(t)|_\infty$, tend to vary less and less. This explains the empirical behavior that we can verify in Figs. 3.11a to 3.11c. Analytically, we can verify that, when the particles start to walk, i.e., when $t = 1$, the maximum variation of $|\bar{N}(t+1) - \bar{N}(t)|_\infty$ is given by:

$$|\bar{N}(1) - \bar{N}(0)|_\infty \leq c \left(\frac{2}{V} - \frac{1}{V+1} \right), \quad (3.31)$$

where c is a real positive constant which depends on the level of competitiveness, which in turn is directly proportional to λ . Such upper limit expression translates the maximum variation that occurs from time $t = 0$ to $t = 1$, which happens when, at $t = 0$, a vertex is not receiving any visits, but, at time $t = 1$, it is being visited by exactly one particle. Generalizing this equation for an arbitrary t , $|\bar{N}(t+1) - \bar{N}(t)|_\infty$ is always bounded by the following expression:

$$|\bar{N}(t+1) - \bar{N}(t)|_\infty \leq c \left(\frac{t+2}{V+t} - \frac{t}{V+t+1} \right), \quad (3.32)$$

which demonstrates that, as the previous case, for any $t \leq \infty$, the proposed model presents fluctuations around a quasi-stationary state. From this analysis, it is clear that the $|\bar{N}(t+1) - \bar{N}(t)|_\infty$ is a much more well-behaved indicator than the $\langle R(t) \rangle$ measure. This explains the reason why we have chosen it as the termination criterion of the proposed algorithm.

In either cases, we can see that the proposed model generally does not converge to a fixed point. Better stated, it only converges to a fixed point in two peculiar situations:

1. When the communities or groups are completely isolated from each other, and each community group exactly owns a single particle;
2. When only the preferential walking rule is applied for the particles, i.e., when $\lambda = 0$.

As we can see, these two special situations cannot be very well established in real situations. The first situation is unfeasible in real-world problems, since communities do overlap and, more, we do not know their localizations prior to the clustering task. Therefore, one could not possibly put exactly a single particle in each community. The second situation is not welcomed since, as we have already emphasized, the model does not provide good clustering results when $\lambda = 0$.

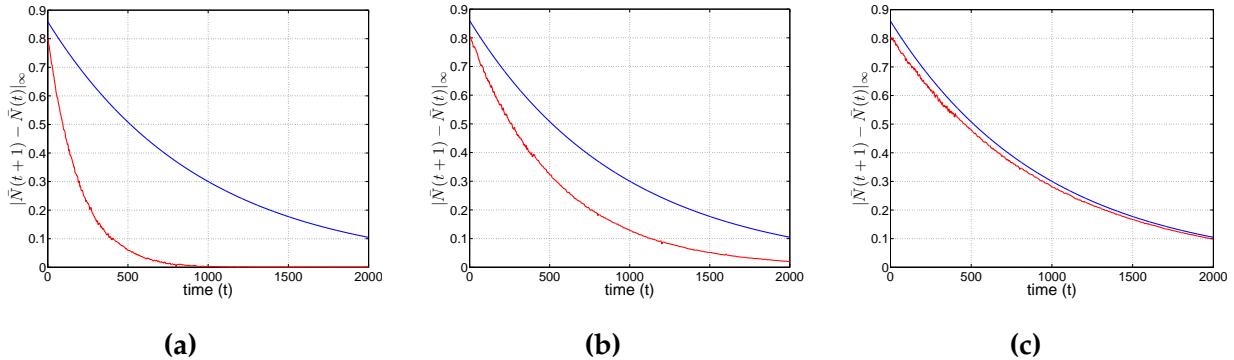


Figure 3.11: Convergence analysis of the particle competition algorithm when $|N(t+1) - N(t)|_\infty$ is used. The upper theoretical limit is shown in the blue curve when $c = K$. The algorithm is run against the binary artificial databases, as one can see in Fig. 3.9. Here, we inspect how the $|N(t+1) - N(t)|_\infty$ measure varies as time progresses. Results obtained on (a) well-posed groups (Fig. 3.9a); (b) somewhat well-posed groups (Fig. 3.9b); (c) not well-posed groups (Fig. 3.9c).

Whenever these two very particular situations are not satisfied, the model instead oscillates within a finite region, whose amplitude depends on the structure of the network. As a consequence, the diameter of this region indirectly depends on the competition level among the particles, which is translated by the value of parameter λ . That is, a wider region corresponds to a more mixed network. In light of that, the proposed model does not present asymptotic stability, but instead it presents structural stability. In our opinion, this characteristic is not a disadvantage, because it is more similar to several real-world systems on account of the noises and other uncontrolled variables presented by the real environment.

Next, we will quantitatively investigate the performance of the elected termination criterion, i.e., $|\bar{N}(t+1) - \bar{N}(t)|_\infty$ in binary random clustered networks, whose construction methodology has been discussed in Section 2.1.4. Table 3.4 shows the accuracy of the model, as well as from which iteration $|\bar{N}(t+1) - \bar{N}(t)|_\infty \leq 0.05$ starts to be satisfied in random clustered networks with varying sizes and mixtures. One can see that the model's performance is size-invariant, since it achieves similar outputs when different network sizes are used. Moreover, we see that the termination criterion takes longer as the mixture of the classes grows, confirming our previous analysis.

3.2.3 Impact Analysis of Different Similarity Functions in the Network Formation Step

In this section, we will analyze the impact of choosing different similarity functions, when constructing the network, on the model's performance. It is worth noting that, on one hand, since the proposed method only relies on the already constructed net-

Table 3.4: Number of iterations to satisfy $|\bar{N}(t+1) - \bar{N}(t)|_{\infty} \leq 0.05$ and the accuracy achieved by the algorithm considering the elected termination criterion in random clustered networks with varying sizes and mixtures.

V	Mixture	Accuracy (%)	Number of Iterations to satisfy $ \bar{N}(t+1) - \bar{N}(t) _{\infty} \leq 0.05$
100	0.1	100.0	381
	0.2	100.0	498
	0.3	100.0	705
	0.4	96.4	1 140
	0.5	88.7	1 962
1 000	0.1	100.0	3 739
	0.2	100.0	5 103
	0.3	100.0	6 877
	0.4	96.6	9 830
	0.5	88.5	16 393
10 000	0.1	100.0	29 984
	0.2	100.0	53 627
	0.3	100.0	73 753
	0.4	96.6	91 847
	0.5	88.6	169 635

work to evolve in time, the inner mechanism of the algorithm is unchanged, because it does not depend on similarity functions, as it derives its decision using the already established network topology. On the other hand, the adjacency matrix of the network will change, because the network construction techniques, such as k -nearest neighbors, do rely on the selected similarity function. In view of that, we are able to conclude that the particle competition algorithm is indirectly impacted by the similarity function on data clustering tasks, because different networks are supplied if one applies different similarity functions on the network construction step. However, in community detection tasks (network is provided), the method does not depend on any similarity functions.

Formally, a similarity measure is defined as a mathematical function which reflects the strength of relationship between two data items. Conversely, a dissimilarity measure deals with the level of divergence between two data items and is often categorized as a distance. Some of the most well-known distances are given in Table 3.5. For a comprehensive list, see ([Cha, 2007](#)). Say that d is the dissimilarity between two data items, then $s = 1 - d$ is defined as their similarity.

Now, with these concepts at hand, we will investigate the cluster detection accuracy of the model when applied to several data sets from the UCI Machine Learning Repository ([Frank and Asuncion, 2010](#)). A meta-description of the considered data sets is supplied in Table 3.6. For a detailed description, one can refer to ([Frank and Asuncion,](#)

Table 3.5: Some well-known dissimilarity (distance) functions. $v = (v_1, \dots, v_d)$ and $w = (w_1, \dots, w_d)$ are d -dimensional data items.

Denomination	Expression
Euclidean distance	$d_{Euc} = v - w ^2$
Cosine distance	$d_{Cos} = \frac{v \cdot w}{ v w }$
Jaccard distance	$d_{Jac} = \frac{v \cdot w}{ v ^2 + w ^2 - v \cdot w}$
Dice distance	$d_{Dic} = \frac{ v - w ^2}{ v ^2 + w ^2}$

2010). Observe that, as we are interested in clustering matters (unsupervised learning), the available information about the class labels has been used only for evaluation of the results (cluster detection accuracy or ground truth).

Table 3.6: Meta-description of the UCI data sets used in the experiments of testing the impact of different similarity functions on the model's performance.

	# Instances	# Dimension	# Classes
Breast Cancer	699	9	2
Glass	214	9	6
Image Segmentation	2 310	20	7
Ionosphere	351	34	2
Iris	150	4	3
Vowel	90	10	11
Wine	178	13	3
Nursery	12 960	9	5

The computer simulations results are reported in Table 3.7 for the UCI data sets described in Table 3.6. One can see that, on one hand, the Euclidean distance has consistently given the best average cluster detection accuracy and the Dice distance seems to provide a smaller standard deviation. On the other hand, the cosine distance did not perform well. It is worth stressing that, while the Euclidean Distance measures the actual distance between the two points of interest (straight line), the cosine can be thought of as measuring their apparent distance as viewed from the origin. Another way to think of cosine similarity is as measuring the relative proportions of the various features or dimensions - when all the dimensions between two vectors are in proportion (correlated), you get maximum similarity. The Euclidean, Jaccard and Dice distances are more concerned with absolutes, which explains why these measures have given a similar cluster detection accuracy. Having in mind that, if the cosine similarity is used in the network construction, the resulting network may not reflect well the raw data, because it may not satisfy the cluster and smoothness assumptions, since it investigates the angle between the data items. Therefore, there could be two data items very far away from each other, but aligned with respect to the origin. In this situation,

the cosine similarity would accuse the objects to be very similar. In clustering tasks, this would seriously flaw the cluster discovery process, since the network would not be contemplating well the reality of the data relationships (Cha, 2007). However, it is worth addressing that this particular characteristic is very welcomed in some situations, such as in natural language processing.

Table 3.7: Clustering accuracy results when different similarity functions are used to construct the network. The results are averaged over 30 runs.

	Euclidean Distance	Cosine Distance	Jaccard Distance	Dice Distance
Breast Cancer	95.90 ± 0.74	86.85 ± 0.89	92.49 ± 0.76	93.22 ± 0.69
Glass	52.92 ± 6.56	44.18 ± 5.88	46.03 ± 6.06	50.46 ± 5.45
Image Segmentation	67.73 ± 1.36	66.73 ± 1.73	67.55 ± 1.58	67.71 ± 1.31
Ionosphere	75.01 ± 2.98	69.93 ± 3.37	73.11 ± 3.19	74.96 ± 2.72
Iris	89.57 ± 2.27	85.84 ± 2.51	88.00 ± 2.55	88.72 ± 2.18
Vowel	47.52 ± 1.45	46.53 ± 1.72	47.55 ± 1.80	48.10 ± 1.22
Wine	74.49 ± 1.62	71.14 ± 1.54	73.73 ± 1.61	73.67 ± 1.51
Nursery	45.41 ± 2.29	44.16 ± 2.51	44.83 ± 2.44	45.46 ± 2.09

3.3 Theoretical Analysis of the Technique

In this section, a mathematical analysis of the competitive system is supplied. Also, we show that the competitive system reviewed in the previous section reduces to multiple independent random walks when a special situation occurs.

3.3.1 Mathematical Analysis

Primarily, it is of utter importance to determine the transition probability function of system ϕ , i.e., $P(X(t+1) | X(t))$, before any rigorous analysis is taken. Note that the marginal probability of the system's state $P(X(t))$ can be written in terms of a joint probability of each of the components of the system's state, meaning $P(X(t)) = P(N(t), p(t), E(t), S(t))$. In fact, using this fact and applying the Product Rule, one has:

$$\begin{aligned}
P(X(t+1) | X(t)) &= P(N(t+1), p(t+1), E(t+1), S(t+1) | N(t), p(t), E(t), S(t)) \\
&= P(S(t+1) | N(t+1), p(t+1), E(t+1), N(t), p(t), E(t), S(t)) \\
&\quad \times P(N(t+1), p(t+1), E(t+1) | N(t), p(t), E(t), S(t)) \\
&= P_{S(t+1)} P(E(t+1) | N(t+1), p(t+1), N(t), p(t), E(t), S(t)) \\
&\quad \times P(N(t+1), p(t+1) | N(t), p(t), E(t), S(t)) \\
&= P_{S(t+1)} P_{E(t+1)} P(N(t+1) | p(t+1), N(t), p(t), E(t), S(t)) \\
&\quad \times P(p(t+1) | N(t), p(t), E(t), S(t)) \\
&= P_{S(t+1)} P_{E(t+1)} P_{N(t+1)} P_{p(t+1)}.
\end{aligned} \tag{3.33}$$

where $P_{S(t+1)} = P(S(t+1) | N(t+1), p(t+1), E(t+1), X(t))$, $P_{E(t+1)} = P(E(t+1) | N(t+1), p(t+1), X(t))$, $P_{N(t+1)} = P(N(t+1) | p(t+1), X(t))$, and $P_{p(t+1)} = P(p(t+1) | X(t))$. Next, the algebraic derivation of these four quantities is explored.

Discovering the $P_{p(t+1)}$ Factor

Observing that the stochastic vector $p(t+1)$ is directly evaluated from $\mathbb{P}_{\text{transition}}(t)$ given in (3.19), which in turn only requires $p(t)$ and $N(t)$ to be constructed ($X(t)$ is given), then the following equivalence holds:

$$P_{p(t+1)} = P(p(t+1) | X(t)) = \mathbb{P}_{\text{transition}}(N(t), p(t)). \tag{3.34}$$

Here, we have used $\mathbb{P}_{\text{transition}}(N(t), p(t))$ to emphasize the dependence of the transition matrix on $N(t)$ and $p(t)$.

Discovering the $P_{N(t+1)}$ Factor

In this case, taking a close look at $P_{N(t+1)} = P(N(t+1) | p(t+1), X(t))$, we can verify that, besides the previous state $X(t)$, known additional information is supplied, which is the knowledge about $p(t+1)$. By a quick analysis of the update rule given in the second expression of system ϕ , it is possible to completely determine $N(t+1)$, since $p(t+1)$ and $N(t)$ are known. Owing to that, the following equation holds:

$$\begin{aligned}
P_{N(t+1)} &= P(N(t+1) | p(t+1), X(t)) \\
&= \mathbb{1}_{\{N(t+1)=N(t)+Q_N(p(t+1))\}},
\end{aligned} \tag{3.35}$$

where $Q_N(p(t+1))$ is a matrix with $\dim(Q_N) = V \times K$ and dependent on $p(t+1)$, whose (i,j) th-entry is given by:

$$Q_N(p(t+1))(i,k) = \mathbb{1}_{\{p^{(k)}(t+1)=i\}}, \quad (3.36)$$

The argument in the indicator function shown in (3.35) is essentially the first expression of system ϕ , but in a matrix notation. In brief, (3.35) will result in 1 if the computation of $N(t+1)$ is correct, given $p(t+1)$ and $N(t)$, i.e., it is in compliance with the dynamical system rules; and 0, otherwise.

Discovering the $P_{E(t+1)}$ Factor

For the third term, $P_{E(t+1)}$, we have knowledge of the previous state $X(t)$, as well as of $p(t+1)$ and $N(t+1)$. By (3.7), we see that $\bar{N}(t+1)$ can be directly calculated from $N(t+1)$, i.e., having knowledge of $N(t+1)$ permits us to evaluate $\bar{N}(t+1)$, which, probabilistically speaking, is also a given information. In light of this, together with (3.13), one can see that $E(t+1)$ can be evaluated if we have information of $E(t)$, $p(t+1)$, and $\bar{N}(t+1)$, which we actually do. On account of that, $P_{E(t+1)}$ can be surely determined and, analogously to the calculation of the $P_{N(t+1)}$, is given by:

$$\begin{aligned} P_{E(t+1)} &= P(E(t+1) | N(t+1), p(t+1), X(t)) \\ &= \mathbb{1}_{\{E(t+1)=E(t)+\Delta \times Q_E(p(t+1), N(t+1))\}}, \end{aligned} \quad (3.37)$$

where $Q_E(p(t+1), N(t+1))$ is a matrix with $\dim(Q_E) = 1 \times K$ and dependence on $N(t+1)$ and $p(t+1)$. The k th-entry, $k \in \mathcal{K}$, of such matrix is calculated as:

$$Q_E^{(k)}(p(t+1), N(t+1)) = \mathbb{1}_{\{\text{owner}(k,t+1)\}} - \mathbb{1}_{\{\neg \text{owner}(k,t+1)\}}. \quad (3.38)$$

Note that the argument of the indicator function in (3.38) is essentially (3.13) in a compact matrix form. Indicator functions were employed to describe the two types of behavior that this variable can have: an increment or decrement of the particle's energy. Suppose that particle $k \in \mathcal{K}$ is visiting a vertex that is being dominated by itself, then only the first indicator function in (3.38) will be enabled; hence, $Q_E^{(k)}(p(t+1), N(t+1)) = 1$. Similarly, if particle k is visiting a vertex that is being dominated by a rival particle, then the second indicator function is enabled, yielding $Q_E^{(k)}(p(t+1), N(t+1)) = -1$. This behavior together with (3.37) is exactly the expression given by (3.13) in a compact matrix form.

Discovering the $P_{S(t+1)}$ Factor

Lastly, for the forth term, $P_{S(t+1)}$, we have knowledge of $E(t+1)$, $N(t+1)$, $p(t+1)$, and the previous internal state $X(t)$. By a quick analysis of (3.14), one can verify that the calculation of the k th-entry of $S(t+1)$ is completely characterized once $E(t+1)$ is known. In this way, one can surely evaluate $P_{S(t+1)}$ in this scenario as follows:

$$\begin{aligned} P_{S(t+1)} &= P(S(t+1) \mid E(t+1), N(t+1), p(t+1), X(t)) \\ &= \mathbb{1}_{\{S(t+1)=Q_S(E(t+1))\}}, \end{aligned} \quad (3.39)$$

where $Q_S(E(t+1))$ is a matrix with $\dim(Q_S) = 1 \times K$ and has dependence on $E(t+1)$. The k th-entry, $k \in \mathcal{K}$, of such matrix is calculated as:

$$Q_S^{(k)}(E(t+1)) = \mathbb{1}_{\{E^{(k)}(t+1)=\omega_{\min}\}}. \quad (3.40)$$

The Transition Probability Function

Substituting (3.34), (3.35), (3.37), and (3.39) into (3.33), we are able to encounter the transition probability function of the competitive dynamical system:

$$\begin{aligned} P(X(t+1) \mid X(t)) &= \mathbb{1}_{\{N(t+1)=N(t)+Q_N(p(t+1))\}} \\ &\quad \times \mathbb{1}_{\{S(t+1)=Q_S(E(t+1))\}} \\ &\quad \times \mathbb{1}_{\{E(t+1)=E(t)+\Delta Q_E(p(t+1), N(t+1))\}} \\ &\quad \times \mathbb{P}_{\text{transition}}(N(t), p(t)) \\ &= \mathbb{1}_{\{\text{Compliance}(t)\}} \mathbb{P}_{\text{transition}}(N(t), p(t)), \end{aligned} \quad (3.41)$$

where $\text{Compliance}(t)$ is a logical expression given by:

$$\begin{aligned} \text{Compliance}(t) &= [N(t+1) = N(t) + Q_N(p(t+1))] \\ &\wedge [S(t+1) = Q_S(E(t+1))] \wedge [E(t+1) = \\ &\quad E(t) + \Delta Q_E(p(t+1), N(t+1))], \end{aligned} \quad (3.42)$$

i.e., $\text{Compliance}(t)$ encompasses all the rules that have to be satisfied in order to all the indicator functions in (3.41) produce 1. If all the values provided to (3.41) are in compliance with the dynamics of the system, then $\text{Compliance}(t) = \text{true}$ and the indicator function $\mathbb{1}_{\{\text{Compliance}(t)\}}$ yields 1; otherwise, if there is at least one measure that does not satisfy the system, then, from (3.42), the chain of logical AND will produce false.

As a consequence, $\text{Compliance}(t) = \text{false}$ and the indicator function $\mathbb{1}_{\{\text{Compliance}(t)\}}$ in (3.41) yields 0, resulting in a zero-valued transition probability.

Discovering the Distribution $P(N(t))$

With the transition probability function derived in the previous section, we now turn our attention to determining the marginal distribution $P(N(t))$ for a sufficient large t . First, the Markovian property of system ϕ is demonstrated as follows.

* * *

Proposition 1. $\{X(t) : t \geq 0\}$ is a Markovian process.

Proof. We seek to infer that system ϕ is completely characterized by only the acquaintance of the present state, i.e., it is independent of all the past states. Having that in mind, the probability expression to make a transition to a specific event X_{t+1} (a set with an element representing an arbitrary next state) in time $t + 1$, given the complete history of the state trajectory, is denoted by:

$$P(X(t+1) \in X_{t+1} | X(t), \dots, X(0)) = \\ P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} | X(t), \dots, X(0)\right). \quad (3.43)$$

Noting that the determination of p_{t+1} only depends on $N(t)$ and $p(t)$, then:

$$P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} | X(t), \dots, X(0)\right) = \\ P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} | X(t)\right) = \\ P(X(t+1) \in X_{t+1} | X(t)). \quad (3.44)$$

Therefore, in view of (3.44), $\{X(t) : t \geq 0\}$ is a Markovian process, since it only depends on the present state to specify the next state and, hence, the past history of the system's trajectory is irrelevant. \square

* * *

The strategy to calculate the distribution $P(N(t))$ is to marginalize the joint distribution of the system's states, i.e., $P(X(0), \dots, X(t))$, with respect to $N(t)$ (a component of $X(t)$). Mathematically, using Proposition 1 on this joint distribution $P(X(0), \dots, X(t))$, we get:

$$\begin{aligned} P(X(0), \dots, X(t)) &= P(X(t) | X(t-1)) \\ &\quad \times P(X(t-1) | X(t-2)) \\ &\quad \times \dots \times P(X(1) | X(0))P(X(0)). \end{aligned} \quad (3.45)$$

Using the transition function that governs system ϕ , as illustrated in (3.41), to each shifted term in (3.45), we get:

$$P(X(0), \dots, X(t)) = P(X(0)) \prod_{u=1}^{t-1} \left[\mathbb{1}_{\{\text{Compliance}(u)\}} \mathbb{P}_{\text{transition}}(N(u), p(u)) \right], \quad (3.46)$$

where $P(X(0)) = P(N(0), p(0), E(0), S(0))$. But, we are interested in knowing the marginal distribution $N(t)$ as $t \rightarrow \infty$. We can obtain it from the joint distribution calculated in (3.46), summing over all the possible values of random variables with no relevance in the analysis, i.e., $N(t-1), \dots, N(0), p(t), \dots, p(0), E(t), \dots, E(0), S(t), \dots, S(0)$. In doing so, it is worth studying the limits of $N(t)$ for an arbitrary t , because the domain that an entry of $N(t)$ can take is $[1, \infty)$. With this study, we expect to find the reachable values of every entry of matrix $N(t)$ for any t . In this way, values which exceed these limits are guaranteed to happen with probability 0. Lemma 1 precisely supplies this analysis.

* * *

Lemma 1. *The maximum reachable value of $N_i^{(k)}(t)$, $\forall (i, k) \in \mathcal{V} \times \mathcal{K}$, $t \in \mathbb{N}$, is:*

$$N_{i_{\max}}^{(k)}(t) = \begin{cases} \left\lceil \frac{t+1}{2} \right\rceil + 1, & \text{if } t \geq 0 \text{ and } a_{ii} = 0 \\ t + 2, & \text{if } t \geq 0 \text{ and } a_{ii} > 0 \end{cases}. \quad (3.47)$$

Proof. The proof is based on encountering the particle's trajectory that increases $N_i^{(k)}(t)$ in the quickest manner. In this situation, we suppose particle k is generated in vertex i ; otherwise, the maximum theoretical value would never be reached in view of the second expression in (3.23). For the sake of clarity, consider two specific cases, both depicted in Fig. 3.12: (i) networks without self-loops and (ii) networks with self-loops.

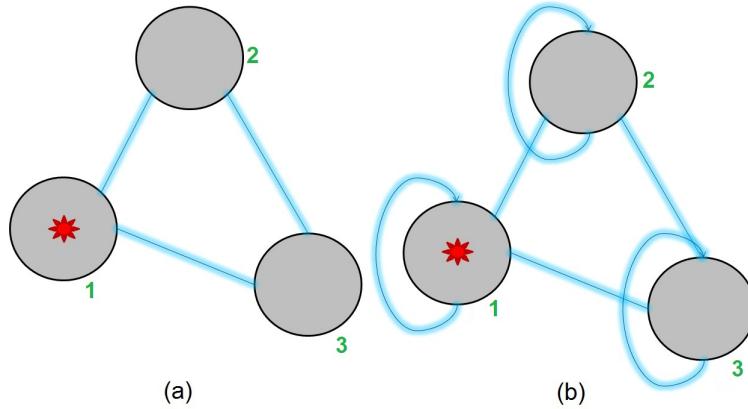


Figure 3.12: An arbitrary network constructed with the purpose of obtaining the largest feasible entry of $N(t)$ for a given t . (a) a network without the presence of self-loops; (b) a network with self-loops.

For the first case, $\forall i \in \mathcal{V} : a_{ii} = 0$. By hypothesis, particle k starts in vertex i at $t = 0$. The quickest manner to increase $N_i^{(k)}(t)$ happens when particle k visits a neighbor of vertex i and immediately returns to vertex i . Repeating this trajectory until time t , $N_i^{(k)}(t)$ precisely matches the first expression in (3.47).

For the second case, $\exists i \in \mathcal{V} : a_{ii} > 0$. By hypothesis, particle k starts in vertex i at $t = 0$. It is clear that the quickest manner to increase $N_i^{(k)}(t)$ occurs when particle k always travels through the self-loop for all t . In view of this, the maximum value that $N_i^{(k)}(t)$ can reach is given by the second expression in (3.47). The “+2” factor appears because the particle initially spawns at vertex i , according to the second expression in (3.23). \square

* * *

In what follows, we analyze the properties of the stochastic vector $E(t)$. The upper limit of the k th-entry of $E(t)$ is always ω_{\max} . Thus, provided that $\omega_{\max} < \infty$, the upper limit is always well-defined. However, this entry does not only accept integer values in-between ω_{\min} and ω_{\max} . Lemma 2 provides all reachable values of $E(t)$ within this interval.

* * *

Lemma 2. *The reachable domain of $E^{(k)}(t), \forall k \in \mathcal{K}, t \in \mathbb{N}$, is:*

$$\begin{aligned} \mathcal{D}_E \triangleq & \left\{ \omega_{\min} + \frac{\omega_{\max} - \omega_{\min}}{K} + n\Delta, n = \{-\lfloor n_i \rfloor, \dots, \lfloor n_m \rfloor\} \right\} \\ & \cup \left\{ \omega_{\min} + n\Delta, n = \left\{ 1, 2, \dots, \left\lfloor \frac{\omega_{\max} - \omega_{\min}}{\Delta} \right\rfloor \right\} \right\} \\ & \cup \left\{ \omega_{\max} - n\Delta, n = \left\{ 1, 2, \dots, \left\lfloor \frac{\omega_{\max} - \omega_{\min}}{\Delta} \right\rfloor \right\} \right\}, \end{aligned} \quad (3.48)$$

where $n_i = \frac{\omega_{\max} - \omega_{\min}}{K\Delta} \geq 0$ and $n_m = \frac{\omega_{\max} - \omega_{\min}}{\Delta} \left(1 - \frac{1}{K}\right) \geq 0$.

Proof. We divide this proof in three main steps, namely the three sets that appear in the expression in the *caput* of this lemma. The first set accounts for supplying all values that are multiples of Δ with the offset $E^{(k)}(0) = \omega_{\min} + \left(\frac{\omega_{\max} - \omega_{\min}}{K}\right)$, $\forall k \in \mathcal{K}$ (see (3.24)). The minimum reachable value is given when $n = n_i$:

$$n_i = \frac{\left(\omega_{\min} + \frac{\omega_{\max} - \omega_{\min}}{K}\right) - \omega_{\min}}{\Delta} = \frac{\omega_{\max} - \omega_{\min}}{K\Delta}, \quad (3.49)$$

whereas the maximum reachable value is given when $n = n_m$:

$$n_m = \frac{\omega_{\max} - \left(\omega_{\min} + \frac{\omega_{\max} - \omega_{\min}}{K}\right)}{\Delta} = \frac{\omega_{\max} - \omega_{\min}}{\Delta} \left(1 - \frac{1}{K}\right). \quad (3.50)$$

After some time, the particle k might reach one of the two possible extremes of energy value: ω_{\min} or ω_{\max} . On account of the $\max(\cdot)$ operator in (3.13), it is also necessary to list all multiple numbers of Δ with these two offsets. The second and third sets precisely fulfill this aspect when the offsets are ω_{\min} and ω_{\max} , respectively. Once the particle enters one of these sets, it never leaves them. Hence, all values have been properly mapped. \square

* * *

Lastly, the upper limit of an arbitrary entry of $S(t)$ is 1, since it is a boolean-valued variable. Observing now that $P(X(0), \dots, X(t)) = P(N(0), p(0), E(0), S(0), \dots, N(t), p(t), E(t), S(t))$, we marginalize this joint distribution with respect to $N(t)$ as follows:

$$P(N(t)) = \sum_{\sim N(t)} P(X(0), \dots, X(t)), \quad (3.51)$$

where $\sim N(t)$ means that we sum over all the possible values of $X(0), \dots, X(t)$, except for $N(t)$ which is inside $X(t) = [N(t) \ p(t) \ E(t) \ S(t)]^T$. Using (3.46) in (3.51), we are able to obtain $P(N(t))$ as follows:

$$P(N(t)) = \sum_{\sim N(t)} \left\{ P(X(0)) \prod_{u=1}^{t-1} \left[\mathbb{1}_{\{\text{Compliance}(u)\}} \mathbb{P}_{\text{trans}}(N(u), p(u)) \right] \right\}. \quad (3.52)$$

Expanding (3.52) using Lemmas 1 and 2, we have:

$$\begin{aligned}
P(N(t)) = & \sum_{p^{(1)}(0)=1}^V \sum_{p^{(2)}(0)=1}^V \dots \sum_{p^{(K)}(0)=1}^V \dots \sum_{p^{(K)}(t)=1}^V \\
& \times \sum_{N_1^{(1)}(0)=1}^{N_{1\max}^{(1)}} \sum_{N_1^{(2)}(0)=1}^{N_{1\max}^{(2)}} \dots \sum_{N_V^{(K)}(0)=1}^{N_{V\max}^{(K)}} \dots \sum_{N_V^{(K)}(t-1)=1}^{N_{V\max}^{(K)}(t-1)} \\
& \times \sum_{E^{(1)}(0) \in \mathcal{D}_E} \sum_{E^{(2)}(0) \in \mathcal{D}_E} \dots \sum_{E^{(K)}(0) \in \mathcal{D}_E} \dots \sum_{E^{(K)}(t) \in \mathcal{D}_E} \\
& \times \sum_{S^{(1)}(0)=0}^1 \sum_{S^{(2)}(0)=0}^1 \dots \sum_{S^{(K)}(0)=0}^1 \dots \sum_{S^{(K)}(t)=0}^1 \\
& \left\{ P(X(0)) \prod_{u=1}^{t-1} \left[\mathbb{1}_{\{\text{Compliance}(u)\}} \mathbb{P}_{\text{trans}}(N(u), p(u)) \right] \right\}. \quad (3.53)
\end{aligned}$$

The summations in the first line of (3.53) account for passing through all possible values of $p(0), \dots, p(t)$. The summations in the second line are responsible for passing through all reachable values of $N(0), \dots, N(t-1)$, where the upper limit is set with the aid of Lemma 1. The third line supplies the summation over all possible values of $E(0), \dots, E(t)$, in which it is utilized the set \mathcal{D}_E defined in Lemma 2. Lastly, the forth line summations cover all the values of $S(0), \dots, S(t)$. Note that the logical expression $\text{Compliance}(u)$ and the transition matrix inside the productory are built up from all these summation indices previously fixed.

* * *

Remark 4. An interesting feature added by this theoretical analysis is that the particle competition model can also accept uncertainty revolving around its initial state, i.e., $P(X(0)) = P(N(0), p(0), E(0), S(0))$. In other terms, the particles' initial locations can be conceptualized as a true distribution itself.

* * *

Discovering the Distribution of the Domination Level Matrix $P(N(t))$

The distribution of the domination level matrix $\bar{N}(t)$ is the fundamental information needed to group up the vertices. First, one can observe that positive integer multiples of $N(t)$ compose the same $\bar{N}(t)$. Therefore, the mapping $N(t) \rightarrow \bar{N}(t)$ is not injective; hence, not invertible. Below, an illustrative example shows this property.

* * *

Example 3. Consider a network with 3 particles and 2 vertices. At time t , suppose that the stochastic process is able to produce two distinct configurations for $N(t)$, as follows:

$$\begin{aligned} N(t) &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}, \\ N'(t) &= \begin{bmatrix} 2 & 2 & 2 \\ 2 & 4 & 6 \end{bmatrix}. \end{aligned} \quad (3.54)$$

Then, the setups in (3.54) applied to (3.7) make clear that both configurations yield the same $\bar{N}(t)$ given by:

$$\bar{N}(t) = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/6 & 1/3 & 1/2 \end{bmatrix}. \quad (3.55)$$

In view of this, the mapping $N(t) \rightarrow \bar{N}(t)$ cannot be injective nor invertible.

* * *

Before proceeding further with the deduction of how to calculate $\bar{N}(t)$ from $N(t)$, let us present some helpful auxiliary results.

* * *

Lemma 3. For any given time t , the following assertions hold $\forall(i, k) \in \mathcal{V} \times \mathcal{K}$:

(a) The minimum value of $\bar{N}_i^{(k)}(t)$ is:

$$\bar{N}_{i_{\min}}^{(k)}(t) = \frac{1}{1 + \sum_{u \in \mathcal{K} \setminus \{k\}} N_{i_{\max}}^{(u)}(t)}. \quad (3.56)$$

(b) The maximum value of $\bar{N}_i^{(k)}(t)$ is:

$$\bar{N}_{i_{\max}}^{(k)}(t) = \frac{N_{i_{\max}}^{(k)}(t)}{N_{i_{\max}}^{(k)}(t) + (K - 1)}. \quad (3.57)$$

Proof. (a) According to (3.7), the minimum value occurs if three conditions are met: (i) particle k is not initially spawned at vertex i ; (ii) particle k never visits vertex i ; and (iii) all other $K - 1$ particles $u \in \mathcal{K} \setminus \{k\}$ visit vertex i in the quickest possible manner, i.e., they follow the trajectory given in Lemma 1. In this way, vertex i will be visited $\sum_{u \in \mathcal{K} \setminus \{k\}} N_{i_{\max}}^{(u)}(t)$ times by the other particles. However, having in mind the initial condition of $N(0)$ shown in the second expression of (3.23), we must add 1 to the total

number of visits received by vertex i . By virtue of that, it is expected that the total number of visits to be $1 + \sum_{u \in \mathcal{K} \setminus \{k\}} N_{i_{\max}}^{(u)}(t)$. In view of this scenario, applying (3.7) to this configuration yields (3.56).

(b) The maximum value happens if three conditions are satisfied: (i) particle k initially spawns at vertex i ; (ii) particle k visits i in the quickest possible manner; and (iii) all other particles $u \in \mathcal{K} \setminus \{k\}$ never visit i . In this scenario, it is hoped that vertex i will receive $N_{i_{\max}}^{(k)}(t) + (K - 1)$ visits, where the second term in the summation is due to the initialization of $N(0)$, as the second expression in (3.23) reveals. This information, together with (3.7), implies (3.57). \square

* * *

Remark 5. If the network does not have self-loops, then (3.56) reduces to:

$$\bar{N}_{i_{\min}}^{(k)}(t) = \frac{1}{1 + (K - 1)N_{i_{\max}}^{(k)}(t)}. \quad (3.58)$$

* * *

The following Lemma provides all reachable elements that an arbitrary entry of $\bar{N}(t)$ can have.

* * *

Lemma 4. Denote num/den as an arbitrary irreducible fraction. Consider that the set \mathcal{I}_t retains all the reachable values of $\bar{N}_i^{(k)}(t)$, $\forall (i, k) \in \mathcal{V} \times \mathcal{K}$, for a fixed t . Then, the elements of \mathcal{I}_t are composed of all elements satisfying the following constraints:

- (a) The minimum element is given by the expression in (3.56).
- (b) The maximum element is given by the expression in (3.57).
- (c) All the irreducible fractions within the interval delimited by (a) and (b) such that:

- I. $\text{num}, \text{den} \in \mathbb{N}^*$;
- II. $\text{num} \leq N_{i_{\max}}^{(k)}(t)$;
- III. $\text{den} \leq \sum_{u \in \mathcal{K}} N_{i_{\max}}^{(u)}(t)$.

Proof. (a) and (b) Straightforward from Lemma 3; (c) Firstly, we need to remember that $N_i^{(k)}(t)$ may only take integer values. According to (3.7), $\bar{N}_i^{(k)}(t) = \text{num}/\text{den}$ is a ratio of integer numbers. As a consequence, num and den must be integers and clause I is

demonstrated. In view of (3.7), num only registers visits performed by a single particle. Therefore, the upper bound of it is established by Lemma 1, i.e., $N_{i_{\max}}^{(k)}(t)$. Hence, clause II is proved. Looking at the same expression, observe that den registers the number of visits performed by all particles. Again, using Lemma 1 proves clause III. \square

* * *

Another interesting feature of the set \mathcal{I}_t is elucidated in the following Lemma.

* * *

Lemma 5. *Given $t \leq \infty$, the set \mathcal{I}_t indicated in Lemma 4 is always finite.*

Proof. In order to demonstrate this lemma, it is enough to verify that each set appearing in the *caput* of Lemma 4 is finite.

(a) are (b) are scalars, hence, they are finite sets. (c) Clause I indicates a lower bound for the numerator and the denominator. Clauses II and III reveal upper bounds for the numerator and denominator, respectively. Also from clause I, it can be inferred that the interval delimited by the lower and upper bounds is discrete. Therefore, the number of irreducible fractions that can be made from these two limits is finite.

As all the sets are finite for any t , since \mathcal{I}_t is the union of all these subsets, it follows that \mathcal{I}_t is also finite for any t . \square

* * *

Lemma 4 supplies the reachable values of an arbitrary entry of $\bar{N}(t)$ by means of the definition of the set \mathcal{I}_t . Next, we simply extend this notion to the space spawned by the matrices $\bar{N}(t)$ with dimensions $V \times K$, in such a way that each entry of it must be an element of \mathcal{I}_t as follows:

$$\mathcal{M}_t \triangleq \{\bar{N} : \bar{N}_i^{(k)}(t) \in \mathcal{I}_t, \forall (i, k) \in \mathcal{V} \times \mathcal{K}\}. \quad (3.59)$$

In light of all these previous consideration, we can now provide a compact way of determining the distribution of $\bar{N}(t)$. Following the aforementioned strategy, $P(\bar{N}(t))$ can be calculated by summing over all multiples of $uN(t)$, $u \in \{1, \dots, t\}$ such that $f(uN(t)) = \bar{N}(t)$, where f is the normalization function defined in (3.7). On account of this, we have:

$$P(\bar{N}(t) = U : U \in \mathcal{M}_t) = \sum_{u=1}^t P(f(uN(t)) = U), \quad (3.60)$$

where the upper limit provided in summation of (3.60) is taken using a conservative approach. Indeed, the probability for events such that $N_i^{(k)}(t) > N_{i_{\max}}^{(k)}(t)$ is zero. By virtue of that, we can stop summing whenever any entry of matrix $uN(t)$ exceeds this value. We have omitted this observation from (3.60) for the sake of clarity.

As $t \rightarrow \infty$, $P(\bar{N}(t))$ provides enough information for grouping the vertices. In this case, they are grouped accordingly to the particle that is imposing the highest domination level. Since the domination level is a continuous stochastic variable, the output of this model is fuzzy.

3.3.2 The Proposed Competitive System Generalizes Multiple Independent Random Walks

In this section, we will demonstrate the following interesting property: when $\lambda = 0$ and $\Delta = 0$, the competitive mechanism among the particles and the reanimation feature are turned off, and the system reduces to multiple independent random walkers (Çinlar, 1975). Conversely, when $\lambda > 0$, the competitive mechanism among the particles is enabled and the combination of random-preferential interacting walks occurs. In this case, the reanimation feature is presented depending on the choice of Δ .

* * *

Proposition 2. *If $\lambda = 0$ and $\Delta = 0$, then system ϕ reduces to the case of multiple independent random walks.*

Proof. It is noteworthy to state that, when $\lambda = 0$, the influence of the matrix denoting the preferential movement, $\mathbb{P}_{\text{pref}}(t)$, is taken away. Indeed, when $\lambda = 0$, the coupling between $N(t)$ and $p(t)$ ceases to exist, because the calculation step of $P_{\text{pref}}(t)$ (responsible for the coupling) can be skipped. Moreover, if $\Delta = 0$, then the particles can never get exhausted. In view of these characteristics, the dynamical system ϕ can be easily described by a traditional Markovian process given by:

$$p(t+1) = p(t)\mathbb{P}_{\text{transition}} \quad (3.61)$$

where $\mathbb{P}_{\text{transition}} = \mathbb{P}_{\text{rand}} \otimes \mathbb{P}_{\text{rand}} \otimes \dots \otimes \mathbb{P}_{\text{rand}}$ and $p(t)$ is an enumerated state encompassing all the particles, as described before. Here, the independence among the particles is demonstrated by showing that the generated $N(t)$ by system ϕ is exactly the same as the one produced by the potential matrix of the Markov Chains Theory (Çinlar, 1975). In other words, $N(t)$ can be implicitly calculated from the stochastic process $\{p(t) : t \geq 0\}$.

First, it is useful to find a closed expression for $N(t)$ in terms of $N(0)$. This can be easily done if we iterate the matrix equation $N(t+1) = N(t) + Q$, where Q is as given

in (3.36). In doing so, we get:

$$N(t) = \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} + \sum_{i=0}^t \begin{bmatrix} \mathbb{1}_{\{p^{(1)}(i)=1\}} & \cdots & \mathbb{1}_{\{p^{(K)}(i)=1\}} \\ \mathbb{1}_{\{p^{(1)}(i)=2\}} & \cdots & \mathbb{1}_{\{p^{(K)}(i)=2\}} \\ \vdots & \ddots & \vdots \\ \mathbb{1}_{\{p^{(1)}(i)=V\}} & \cdots & \mathbb{1}_{\{p^{(K)}(i)=V\}} \end{bmatrix}. \quad (3.62)$$

Since this process is stochastic, it is worth determining the expectation of the number of visits $N(t)$ given the particle's initial location $p(0)$. Noting that $\mathbb{E}[\mathbb{1}_{\{A\}}] = P(A)$, we have:

$$\mathbb{E}[N(t) | p(0)] = \begin{bmatrix} 1 & \cdots & 1 \\ 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} + \sum_{i=0}^t \begin{bmatrix} P^i(p_1(0), 1) & \cdots & P^i(p_K(0), 1) \\ P^i(p_1(0), 2) & \cdots & P^i(p_K(0), 2) \\ \vdots & \ddots & \vdots \\ P^i(p_1(0), V) & \cdots & P^i(p_K(0), V) \end{bmatrix}, \quad (3.63)$$

where $P^i(p_j(0), 1)$ denotes the $(p_j(0), 1)$ -entry of $\mathbb{P}_{\text{transition}}$ to the i th-power. But, from the Markov Chains Theory, we have that the so-called truncated potential matrix ([Çinlar, 1975](#)) is given by:

$$R_t(v, k) \triangleq \sum_{i=0}^t \mathbb{P}_{\text{transition}}^i(v, k). \quad (3.64)$$

By virtue of (3.64), each entry of the matrix equation in (3.63) can be rewritten as:

$$\mathbb{E}[N_i^{(j)}(t) | p(0)] = 1 + R_t(p_j(0), i). \quad (3.65)$$

From (3.65), we can infer that each particle does perform an independent random walk according to a Markov Chain. Thus, we are able to conclude that, for $\lambda = 0$ and $\Delta = 0$, all the states of system ϕ follow a traditional Markov Chain process, except for a constant, as demonstrated in (3.65). \square

* * *

The Proposition 2 states that system ϕ reduces to the case of multiple random walks when $\lambda = 0$ and $\Delta = 0$, i.e., we could think that there is a blind competition among the participants. Alternatively, when $0 < \lambda \leq 1$, some orientation is given to the participants, in the sense of defending their territory and not only keep adventuring through the network with no strategy at all. In either case, the reanimation procedure is enabled depending on the choice of Δ .

3.3.3 A Numerical Example

For the sake of clarity, we provide here how to use the theoretical results supplied in the previous section. We limit the demonstration for a single iteration, which is the transition from $t = 0$ to $t = 1$. In order to do so, a simple example composed of a trivial 3-vertex regular network, identical to the one in Fig. 3.12a is analyzed. For the referred example, suppose there are $K = 2$ particles into the network, i.e., $\mathcal{K} = \{1, 2\}$. Let the particle 1 be spawned at vertex 1, and the particle 2, at vertex 2. Suppose also that we have a certainty about the initial locations of the particles at $t = 0$, i.e.:

$$P(X(0)) = P\left(N(0) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}, p(0) = [1 \ 2], E(0), S(0)\right) = 1, \quad (3.66)$$

i.e., there is an 100% chance (certainty) that the particles 1 and 2 will be generated at vertices 1 and 2, respectively. Observe that $N(0)$, $E(0)$, and $S(0)$ are chosen such as to satisfy (3.23), (3.24), and (3.25), respectively. Otherwise, the probability should be 0, in view of (3.41). It is worth to emphasize that the competitive model accepts uncertainty about the initial location of the particles, in a way that we could specify different probabilities to each particle to spawn at different locations. This characteristic is not present in (Quiles *et al.*, 2008), in which it must be fixed a certain position for each particle.

From Fig. 3.12a we can deduce the adjacency matrix A of the graph and, therefore, determine the transition matrix associated to the random movement term for one particle (which is the same to the other particle). Then, applying (3.2), we get:

$$\mathbb{P}_{\text{rand}} = \begin{bmatrix} 0 & 0.50 & 0.50 \\ 0.50 & 0 & 0.50 \\ 0.50 & 0.50 & 0 \end{bmatrix}. \quad (3.67)$$

Given $N(0)$, we can readily establish $\bar{N}(0)$ with the aid of (3.7):

$$\bar{N}(0) = \begin{bmatrix} 0.67 & 0.33 \\ 0.33 & 0.67 \\ 0.50 & 0.50 \end{bmatrix}. \quad (3.68)$$

Using (3.8) we are able to calculate the matrices associated to the preferential movement policy for each particle in the network as:

$$\mathbb{P}_{\text{pref}}^{(1)}(0) = \begin{bmatrix} 0 & 0.40 & 0.60 \\ 0.57 & 0 & 0.43 \\ 0.67 & 0.33 & 0 \end{bmatrix}, \quad (3.69)$$

$$\mathbb{P}_{\text{pref}}^{(2)}(0) = \begin{bmatrix} 0 & 0.57 & 0.43 \\ 0.40 & 0 & 0.60 \\ 0.33 & 0.67 & 0 \end{bmatrix}. \quad (3.70)$$

In order to ease the calculations, let us take $\lambda = 1$, so that (3.19) reduces to $\mathbb{P}_{\text{transition}}(0) = \mathbb{P}_{\text{pref}}^{(1)}(0) \otimes \mathbb{P}_{\text{pref}}^{(2)}(0)$ at time 0, which will be a matrix with dimensions 9×9 given by:

$$\mathbb{P}_{\text{transition}}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.228 & 0.172 & 0 & 0.342 & 0.258 \\ 0 & 0 & 0 & 0.160 & 0 & 0.240 & 0.240 & 0 & 0.360 \\ 0 & 0 & 0 & 0.132 & 0.268 & 0 & 0.198 & 0.402 & 0 \\ 0 & 0.325 & 0.245 & 0 & 0 & 0 & 0 & 0.245 & 0.185 \\ 0.228 & 0 & 0.342 & 0 & 0 & 0 & 0.172 & 0 & 0.258 \\ 0.188 & 0.382 & 0 & 0 & 0 & 0 & 0.142 & 0.288 & 0 \\ 0 & 0.382 & 0.288 & 0 & 0.188 & 0.142 & 0 & 0 & 0 \\ 0.268 & 0 & 0.402 & 0.132 & 0 & 0.198 & 0 & 0 & 0 \\ 0.221 & 0.449 & 0 & 0.109 & 0.221 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3.71)$$

Since in the initial condition depicted in (3.66) the particles 1 and 2 start out at vertices 1 and 2, respectively, the enumerated scalar state for the matters of calculating $p(t+1)$ will be $(1,2) \rightarrow 2$. Hence, we turn our attention to the second row of $\mathbb{P}_{\text{transition}}(0)$ which will completely characterize the transition probabilities for the next state of the dynamical system. A quick analysis of it shows that, out of the 9 possible “next states” of the system, only 4 are plausible (the remaining states have probability 0 to be reached), which are:

$$P \left(N(1) = \begin{bmatrix} 2 & 2 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}, p(1) = \begin{bmatrix} 2 & 1 \end{bmatrix}, E(1), S(1) \mid X(0) \right) = 0.160, \quad (3.72)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 1 \\ 2 & 2 \\ 1 & 2 \end{bmatrix}, p(1) = [2 \ 3], E(1), S(1) \mid X(0) \right) = 0.240, \quad (3.73)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 2 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}, p(1) = [3 \ 1], E(1), S(1) \mid X(0) \right) = 0.240, \quad (3.74)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 2 & 2 \end{bmatrix}, p(1) = [3 \ 3], E(1), S(1) \mid X(0) \right) = 0.360, \quad (3.75)$$

where $X(0)$ is as given in (3.66). Equations (3.72)-(3.75) match our intuition if we take a careful look at Fig. 3.12a: self-looping is not allowed, so the state space that is probabilistically possible of happening can only be these previous 4 states. In other terms, starting from vertex 1, there are only two different choices that the particle can make: either visit vertex 2 or 3. The same reasoning can be applied when we start at vertex 2. Since it is a joint distribution, we multiply these factors, which totalizes 4 different states. Furthermore, as we have fixed $\lambda = 1$, it is expected that the transition to be heavily dependent on the domination levels of the neighboring vertices. In this case, strongly dominated vertices constitute repulsive forces that act against rival particles. With this respect, the preferential or biased behavior of these particles will never adventure in these type of vertices. This is exactly symbolized in (3.75) (denotes the transition probability $(1, 2) \rightarrow (3, 3)$), which possess the highest probability of happening, in account of vertex 3 to be a neutral vertex at time 0, as opposed to the remaining two vertices.

* * *

Remark 6. Alternatively, we could have used the collection of two matrices 3×3 , as given in (3.69) and (3.70) with no loss of generality. Here, we clarify this concept by calculating a single entry of $\mathbb{P}_{\text{transition}}(0)$ using this methodology. Consider we are to calculate the probability according to (3.72), i.e., particle 1 performs a transition from vertex 1 to vertex 2 and particle 2 executes a transition from vertex 2 to vertex 1. For the former case, according to the particle 1's transition matrix (see (3.69)) we have $\mathbb{P}_{\text{pref}}^{(1)}(0)(1, 2) = 0.40$. Likewise, for the last case (see (3.70)), we have $\mathbb{P}_{\text{pref}}^{(2)}(0)(2, 1) = 0.40$. Remembering that $p(0) = [1 \ 2]$ in a scalar form corresponds to the second state of $\mathbb{P}_{\text{transition}}$ and $p(1) = [2 \ 1]$ corresponds to the forth state,

then $\mathbb{P}_{\text{transition}}(0)(2,4) = \mathbb{P}_{\text{pref}}^{(1)}(0)(1,2) \times \mathbb{P}_{\text{pref}}^{(2)}(0)(2,1) = 0.40 \times 0.40 = 0.16$, which is equal to the corresponding entry of the matrix in (3.71).

* * *

Before doing the calculation of the marginal distribution $P(N(1))$, we are required to fix an upper limit for an arbitrary entry of the matrix $N(1)$. This is readily evaluated from (3.47), which results in $N_{i_{\max}}^{(k)}(1) = N_{j_{\max}}^{(k)}(1) = 2, \forall (i,k) \in \mathcal{V} \times \mathcal{K}$, implying that we are only needed to take all numerical combinations for the matrix $N(0)$ such that each entry may only take the values $\{1,2\}$, since larger values will yield probability 0 according to Lemma 1. Moreover, we need to iterate through every feasible value of every entry of $E(0)$ and $E(1)$. In order to do so, we fix $\Delta = 0.25$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. With that, we are able to make use of Lemma 2, which yields $E(t) \in \{0, 0.25, 0.5, 0.75, 1\}$. The limits of the remaining system variables $S(0)$ and $S(1)$ are straightforward. In the present conditions, we have enough information to calculate the marginal distribution $P(N(1))$, according to (3.53):

$$P \left(N(1) = \begin{bmatrix} 2 & 2 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \right) = 1 \times 0.160 = 0.160, \quad (3.76)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 1 \\ 2 & 2 \\ 1 & 2 \end{bmatrix} \right) = 1 \times 0.240 = 0.240, \quad (3.77)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 2 \\ 1 & 2 \\ 2 & 1 \end{bmatrix} \right) = 1 \times 0.240 = 0.240, \quad (3.78)$$

$$P \left(N(1) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 2 & 2 \end{bmatrix} \right) = 1 \times 0.360 = 0.360. \quad (3.79)$$

As the last goal, our task is to determine the distribution $P(\bar{N}(1))$. According to the specified steps in the previous section, we need to find all irreducible fractions that lie within the interval $[0, 1]$ with the constraints derived in the previous section. This means that we only have to consider entries of matrix $\bar{N}(t)$ that contain elements of \mathcal{I} ; the remainder $\bar{N}(t)$ are infeasible and, thus, occur with probability 0. In view of the constraints previously enumerated, $\mathcal{I} = \{1/4, 1/3, 1/2, 2/3, 3/4\}$. Observing that we have the complete distribution of $N(1)$, it is an easy task to apply (3.60), as follows:

$$P \left(\bar{N}(1) = \begin{bmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \\ 1/2 & 1/2 \end{bmatrix} \right) = 0.160, \quad (3.80)$$

$$P \left(\bar{N}(1) = \begin{bmatrix} 2/3 & 1/3 \\ 1/2 & 1/2 \\ 1/3 & 2/3 \end{bmatrix} \right) = 0.240, \quad (3.81)$$

$$P \left(\bar{N}(1) = \begin{bmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \\ 2/3 & 1/3 \end{bmatrix} \right) = 0.240, \quad (3.82)$$

$$P \left(\bar{N}(1) = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \\ 1/2 & 1/2 \end{bmatrix} \right) = 0.360. \quad (3.83)$$

It is noteworthy to reinforce that the mapping between the probabilities of $N(t)$ and $\bar{N}(t)$ is not bijective: in this special simple case that we are studying, we did not have distinct $N(t)$ that could generate the same $\bar{N}(t)$, but as t increases, this is likely to happen quite frequently. This process is repeated until a sufficiently large t or until the system converges to a quasi-stationary state of $\bar{N}(t)$, as discussed in the empirical section.

3.4 Computer Simulations

In this section, we present simulation results in order to assess the effectiveness of the proposed competitive model. Unless specified otherwise, for all simulations hereon, Table 3.8 outlines the parameters used by our technique.

Table 3.8: Standard parameters used by the proposed technique.

Parameter	Value
λ	0.6
Δ	0.07
ω_{\min}	0
ω_{\max}	1
ϵ	0.05

3.4.1 Computational Complexity Analysis

In this section, some experimental tests are performed to verify the time complexity analysis developed in Section 3.1.6. Here we suppose that a network is already given, i.e., Step 2 is not executed. Moreover, we consider that the number of clusters is far less than the quantity of data items, i.e., $K \ll V$. Without considering Step 2, we divide the simulations into two cases: (i) using networks with a constant $\langle k \rangle \ll V$, in which the algorithm is expected to run in linear time order ($\mathcal{O}(V)$); and (ii) using networks with $\langle k \rangle = 0.15V$, i.e., $\langle k \rangle$ proportional to the number of vertices V . In this case, we expect our algorithm to run in quadratic time order ($\mathcal{O}(V^2)$).

In order to conduct our analysis, the Girvan and Newman's benchmark will be employed (Danon *et al.*, 2005), which has been studied in Section 2.1.4. A set of random clustered networks is generated with increasing network sizes ($V = \{1\,000, 2\,000, \dots, 10\,000\}$). Using these networks, the community mixture $z_{\text{out}}/\langle k \rangle$ is incremented, and the community detection accuracy is registered. The time is quantified in an Intel Core 2 CPU 6700 with 4GB of RAM. We compare the time consumed by our method and by the modularity algorithm, which has become a benchmark in the literature. The results in the case of $\langle k \rangle \ll V$ are shown in Figs. 3.13a and 3.13b with medium ($z_{\text{out}}/\langle k \rangle = 0.2$) and high ($z_{\text{out}}/\langle k \rangle = 0.4$) community interconnections, respectively. By inspecting these figures, with regard to our technique, we notice that the processing time increases linearly as a function of the network size. On the other hand, Figs. 3.13c and 3.13d indicate the consumed time in the case of $\langle k \rangle = 0.15V$ with medium ($z_{\text{out}}/\langle k \rangle = 0.2$) and high ($z_{\text{out}}/\langle k \rangle = 0.4$) community interconnections, respectively. In this case, the processing time of both techniques is a quadratic function of the network size. In brief, both behaviors empirically confirm our analysis.

3.4.2 Simulations on Artificial Data

In this section, we provide simulations performed on simple examples with the purpose of illustrating how the proposed algorithm behaves.

In the first experiment, we study the temporal evolution of matrix $\bar{N}(t)$ for a network consisting of $V = 11$ vertices split into 2 unbalanced communities, as depicted in Fig. 3.14. $K = 2$ particles are inserted into the network at the initial positions $p(0) = [2\ 3]$. Figure 3.15a shows the evolitional behavior of the average domination level $\bar{N}(t)$ imposed by particle 1 on vertices 1 to 4 (representing the cyan or light gray community in Fig. 3.14) and on vertices 5 to 11 (representing the red or dark gray community in Fig. 3.14). The same quantities are calculated for particle 2 and are shown in Fig. 3.15b. Note that, as t increases, the average domination level $\bar{N}(t)$ splits into two groups, which correctly identify the 2 communities of the given network. Specifically, from Fig. 3.15a, we can verify that particle 1 dominates vertices 1 to 4, due to the fact

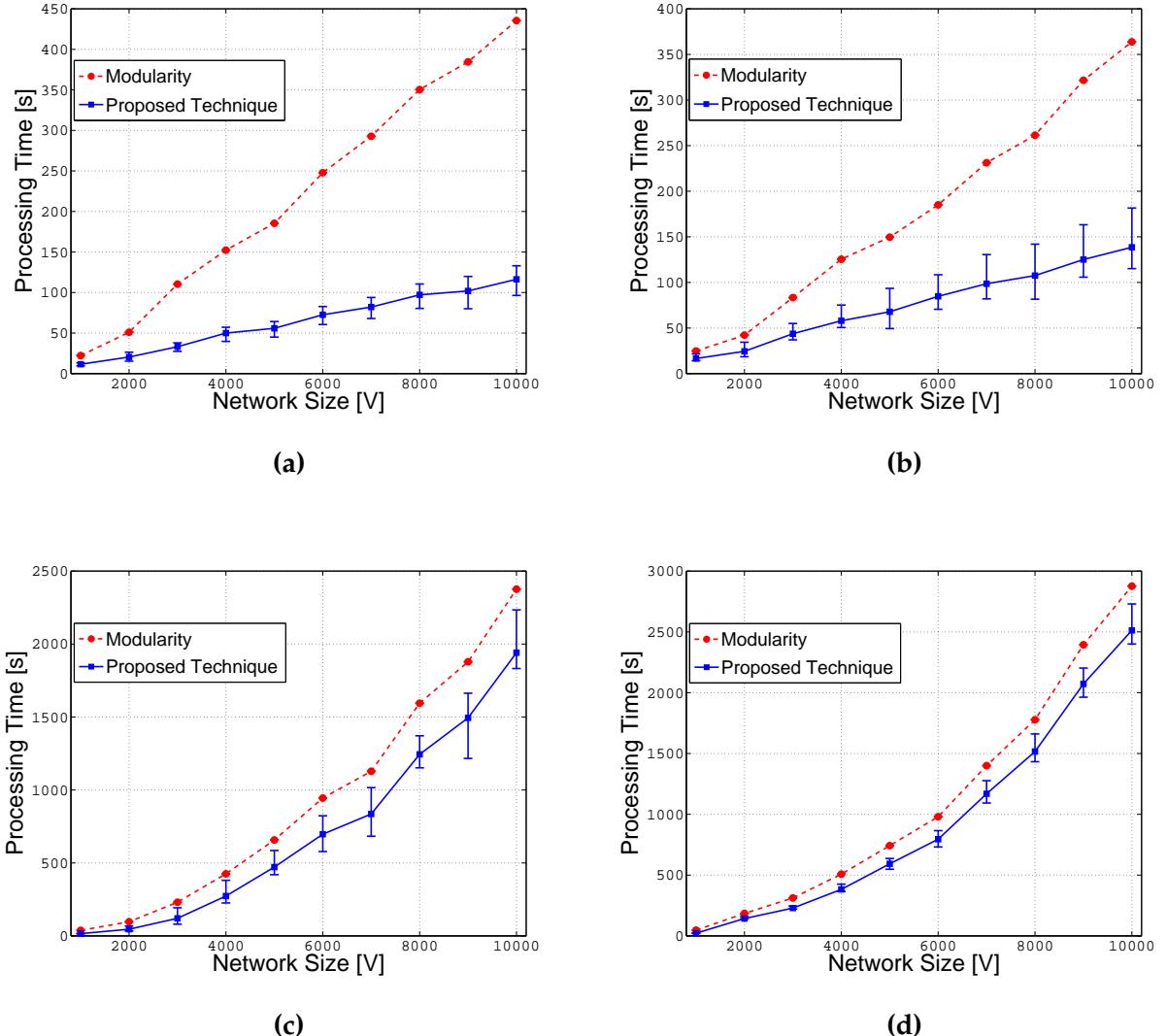


Figure 3.13: Comparison of the consumed time of the proposed technique and the Modularity algorithm. $\langle k \rangle = 16$ in (a) and (b), whereas $\langle k \rangle = 0.15V$ in (c) and (d). $M = 4$ communities. (a) and (c) $z_{\text{out}}/\langle k \rangle = 0.2$. (b) and (d) $z_{\text{out}}/\langle k \rangle = 0.4$. Each point in the trace is averaged over 20 realizations. The error bars represent the maximum and minimum values.

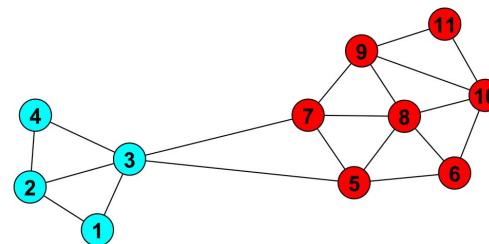


Figure 3.14: A simple network with 11 vertices distributed into 2 unbalanced communities. Vertices 1 to 4 compose a community (cyan or light gray) and vertices 5 to 11 comprise another community (red or dark gray).

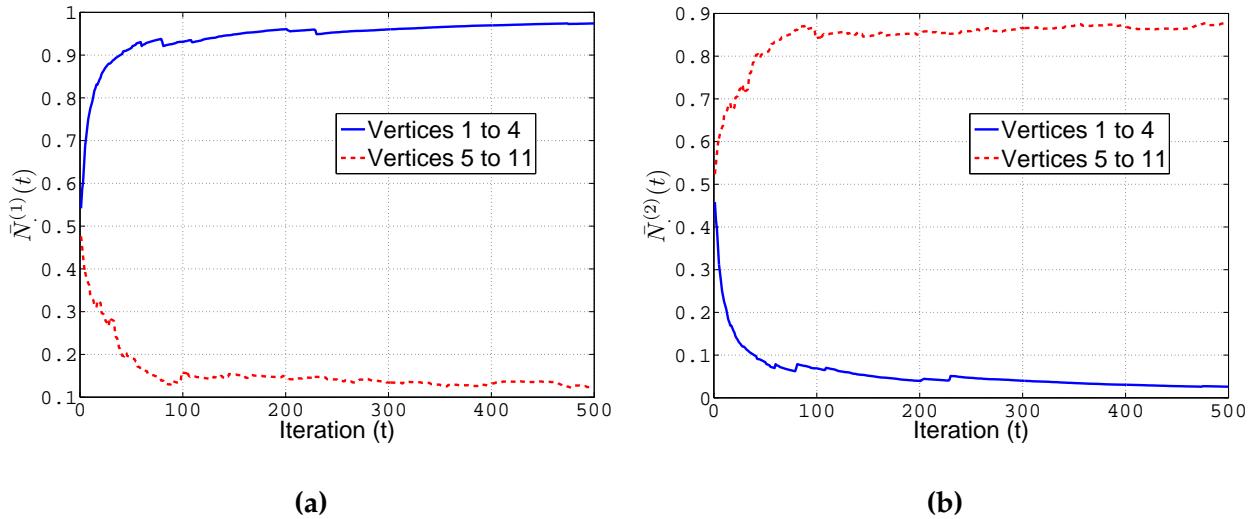


Figure 3.15: Behavior of $\bar{N}(t)$ as the system progresses in time (t). The input network is depicted in Fig. 3.14. (a) Average domination levels of particle 1 on vertices 1 to 4 and on vertices 5 to 11; (b) Average domination levels of particle 2 on the same group of vertices.

that the average domination level on these vertices approaches 1, whereas the average domination level of particle 1 on vertices 5 to 11 decays to 0. Considering Fig. 3.15b, we can use the same logic to confirm that particle 2 dominates vertices 5 to 11.

As our next introductory simulation, we investigate the technique's dependency on the particles' initial locations for a community detection task. Consider Fig. 3.16a, where there are clearly $M = 4$ communities. The black stars represent vertices that do not have an owner. $K = 4$ particles are inserted in a random manner (following a uniform distribution), namely the red or circle-shaped, the blue or square-shaped, the green or diamond-shaped, and the magenta or triangle-shaped particles. The ownership of a vertex is given by its color or shape. Figure 3.16b shows the ownership of the vertices after 300 iterations, Fig. 3.16c, after 800 iterations, and Fig. 3.16d, after 1700 iterations, when all 4 communities have been properly discovered. In order to see the outcome of the algorithm for the worst possible scenario, consider Fig. 3.17a, where we have purposefully inserted all the particles in the same community. The network used is the same as the previous simulation. Figure 3.17b illustrates the ownership of the vertices after 300 iterations, Fig. 3.17c, after 800 iterations, and Fig. 3.17d, after 2000 iterations. One can see that the algorithm again can correctly identify all the 4 communities. These simulations confirm that the community detection performance of the proposed technique does not depend on the initial locations of the particles. This feature is due to the intuitive fact that the particles separate in time as a result of the competition and reanimation mechanisms, even if they are put together at the beginning.

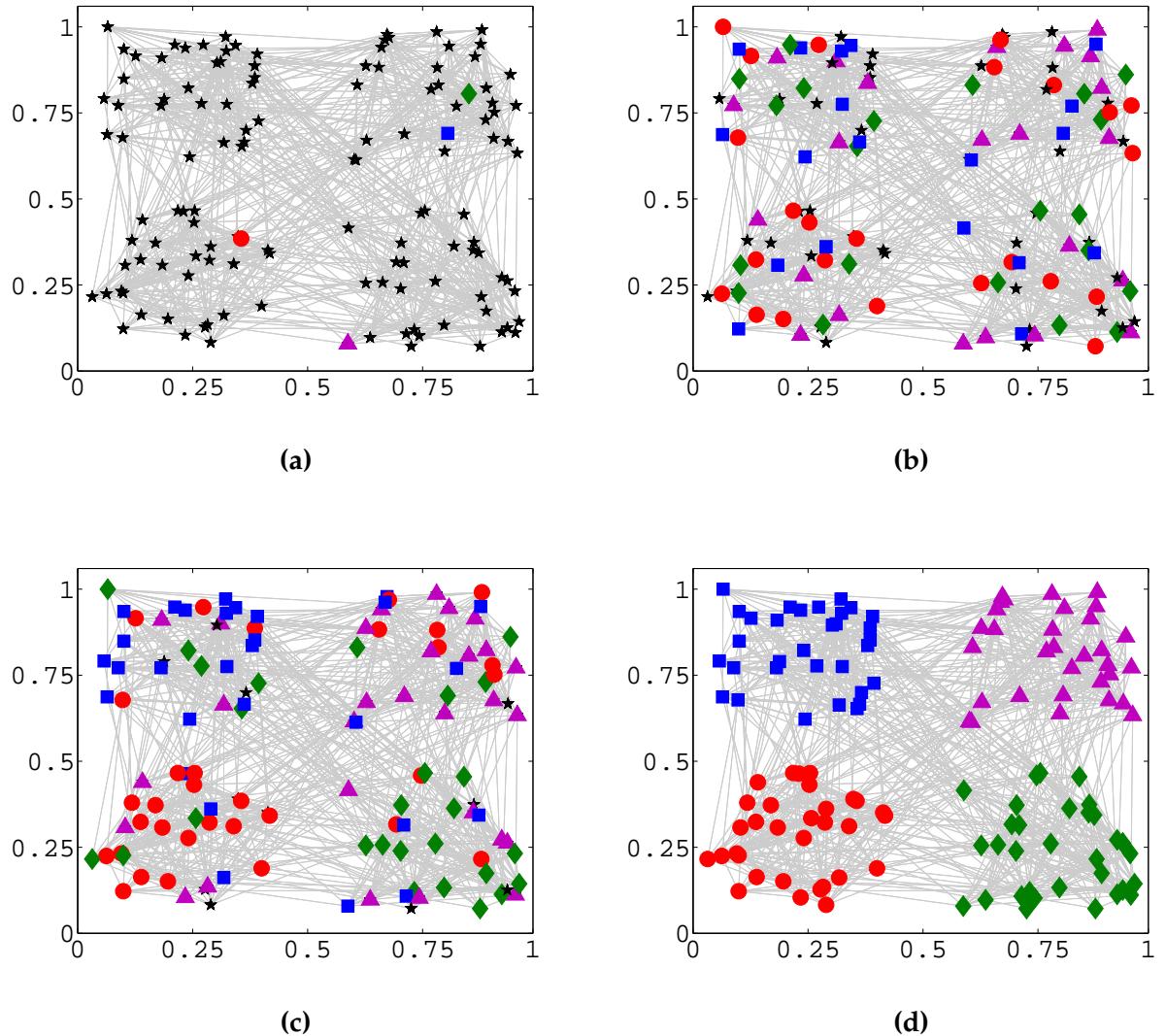


Figure 3.16: Illustration of an artificial community detection process via particle competition. The total number of vertices is $V = 128$, $M = 4$ equally sized communities, $\langle k \rangle = 16$, and $z_{\text{out}}/\langle k \rangle = 0.3$. $K = 4$ particles are inserted in a random manner. Snapshot of the network when: (a) $t = 0$; (b) $t = 300$; (c) $t = 800$; and (d) $t = 1700$.

3.4.3 Simulations for Community Detection on Artificial and Real-World Networks

In this section, we evaluate the performance of the proposed particle competition algorithm on networked data, i.e., the network formation step is not processed. For this end, we make use of two widely employed benchmarks for evaluating the robustness of community detection algorithms: the Girvan and Newman's (GN) benchmark ([Danon et al., 2005](#)) and the benchmark of Lancichinetti *et. al.* ([Lancichinetti et al., 2008](#)), both of which were introduced in Section 2.1.4.

With respect to the GN's benchmark, Fig. 3.18 shows this analysis. Observe that

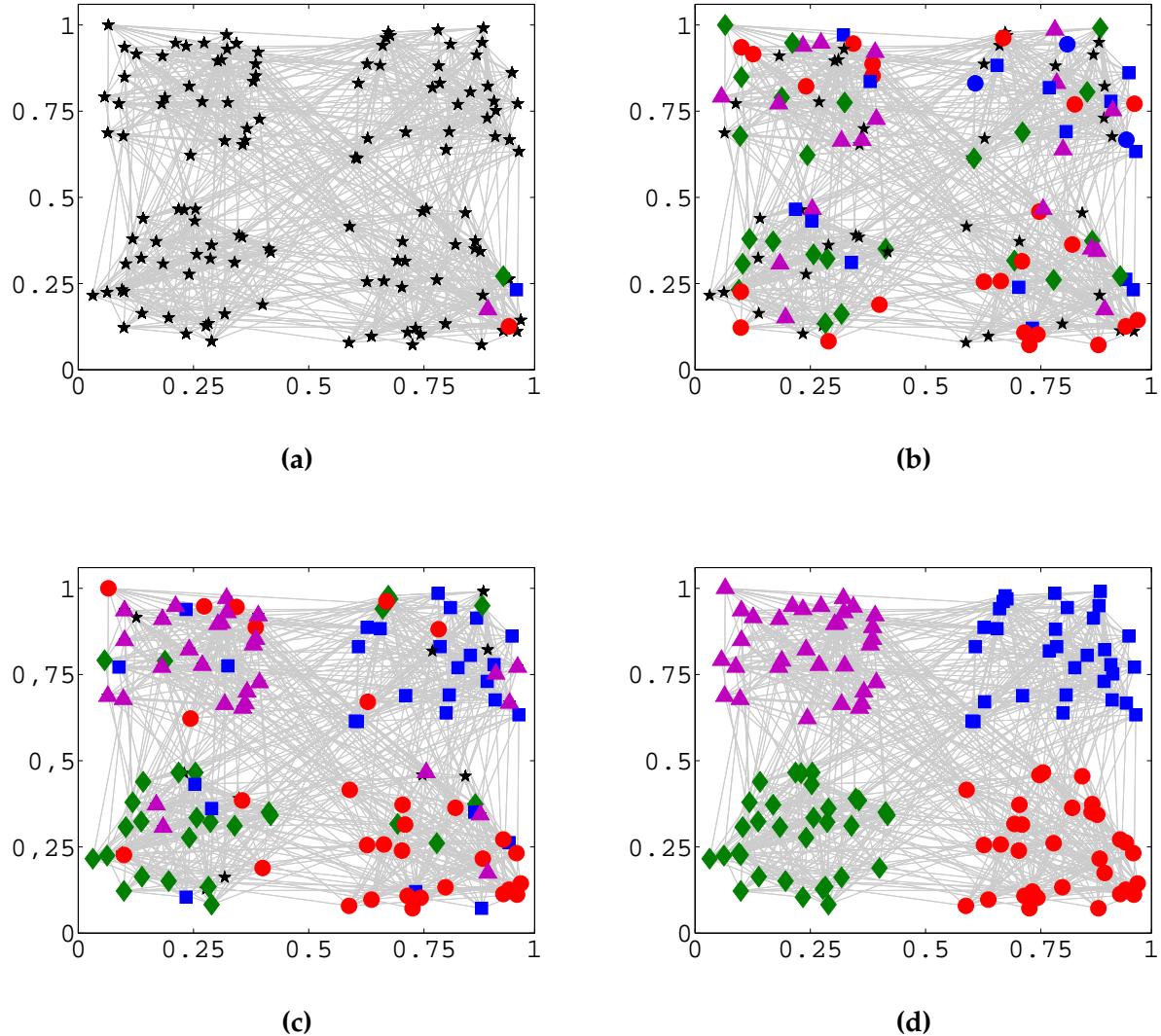


Figure 3.17: Illustration of an artificial community detection process via particle competition. The total number of vertices is $V = 128$, $M = 4$ equally sized communities, $\langle k \rangle = 16$, and $z_{\text{out}}/\langle k \rangle = 0.3$. $K = 4$ particles are purposefully inserted in the same community. Snapshot of the network when: (a) $t = 0$; (b) $t = 300$; (c) $t = 800$; and (d) $t = 2000$.

the proposed technique reaches high community detection rates, even for high values of $z_{\text{out}}/\langle k \rangle$ (highly mixed communities). Furthermore, from a comparison between the results shown in Fig. 2 of (Danon *et al.*, 2005), where it depicts the outcome of 9 different representative community detection techniques with the same experimental setup, and our result shown in Fig. 3.18, one can verify that our technique has outperformed the betweenness, extremal optimization, and modularity techniques (shown in Table 3.1), among other techniques not discussed here. Moreover, if we compare the proposed technique using the same benchmark with the method in (Quiles *et al.*, 2008) (see Fig. 8 of (Quiles *et al.*, 2008)), which is a closely related work, we can also conclude that our technique has obtained better results. In brief, we are able to infer that the

proposed particle competition method has certainly achieved one of the highest community detection rates among all the techniques under comparison with regard to the GN's benchmark.

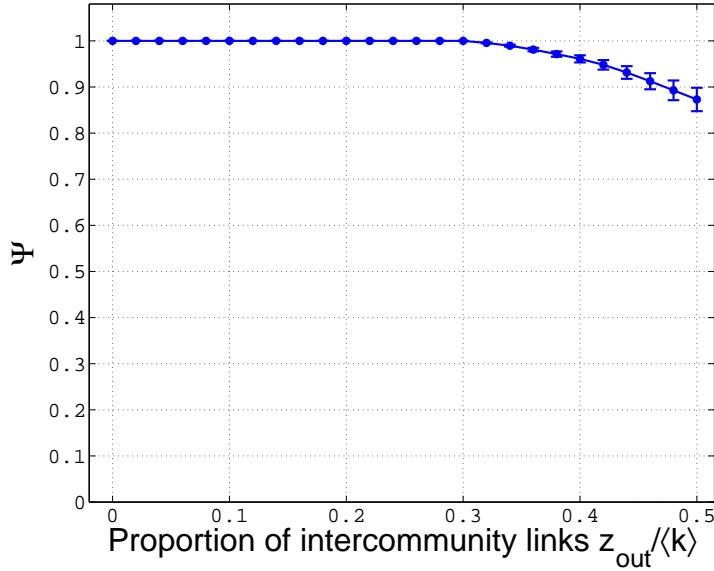


Figure 3.18: GN's community detection benchmark results obtained by the proposed technique. The constructed networks' configurations are: $V = 128$, $M = 4$ equally sized communities, and $\langle k \rangle = 16$. For each point in the trajectory, 200 independent runs are performed. The error bars indicate the standard deviations.

In what concerns the second benchmark, the benchmark of Lancichinetti *et al.*, we fix $\gamma = 2$ in all the related simulations. Figures 3.19a and 3.19b reveal the outcome of the modularity technique, the closely related technique developed in (Quiles *et al.*, 2008), and the proposed technique when we utilize $\beta = 1$ and $\beta = 2$, respectively. The threshold $\mu = 0.5$ (dashed vertical line in the plot) marks the border beyond which communities are no longer defined in a strong sense, which means that each vertex has more neighbors in other communities than in its own (Fortunato, 2010). Overall, our technique has obtained better results than the modularity algorithm and the technique defined in (Quiles *et al.*, 2008). A salient feature presented in the proposed scheme, but not implemented in (Quiles *et al.*, 2008), is the mixture of random and preferential movement that each active particle performs during the same time step. In other words, the resulting transition distribution is given by a convex combination of these two movement types, as (3.1) reveals. On the other hand, in (Quiles *et al.*, 2008), a particle is only allowed to make either a random or a preferential walk during each iteration. When the mixture parameter μ reaches high values, the technique developed in (Quiles *et al.*, 2008) starts to fail in correctly detecting communities. Therefore, we can infer that this probabilistic mixture of random-preferential walks is more robust.

Next, we apply our technique to a real-world data set named Zachary's "karate

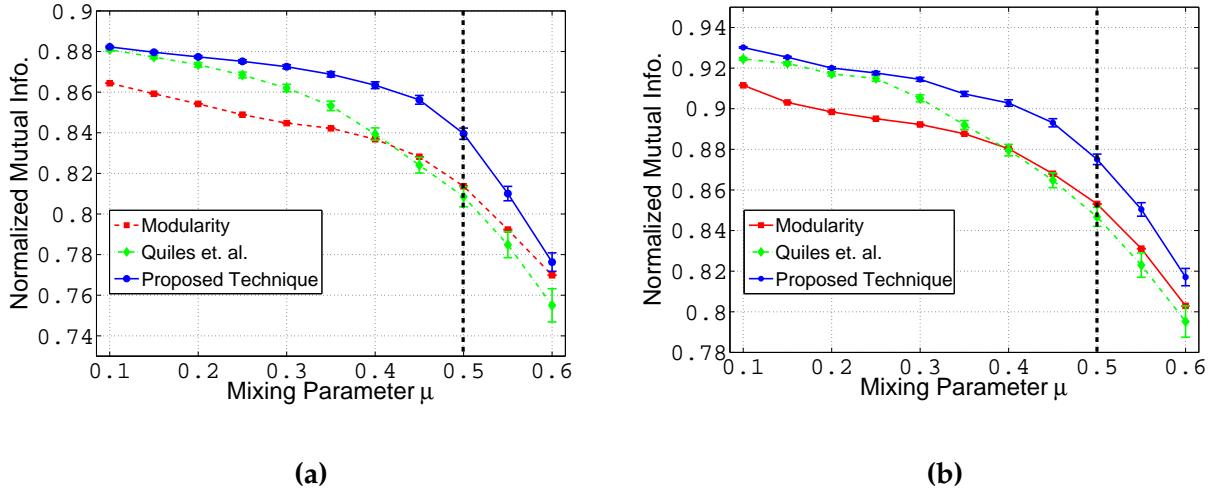


Figure 3.19: Comparison of three community detection algorithms using the benchmark of Lancichinetti *et. al.* The modularity method, the algorithm developed in Quiles *et. al.* (cf. (Quiles *et al.*, 2008)), and the proposed technique are utilized. The constructed networks' configurations are: $V = 10\,000$ and $\langle k \rangle = 15$. $\gamma = 2$. (a) $\beta = 1$. (b) $\beta = 2$. For each point in the trajectory, 30 independent runs are performed. The error bars indicate the standard deviations.

club” network (Zachary, 1977). This is a well-known network from the social science literature, which has become a benchmark test for community detection algorithms. This network exhibits the pattern of friendship among 34 members of a club at an American University in the 1970s. The members are represented by vertices and an edge exists between two members if they know each other. Shortly after the formation of the network, the club dismembered in two as the consequence of an internal dispute, making it an interesting problem for detecting communities. Figure 3.20 shows the outcome of the simulation. The red (dark gray) and blue (gray) colors denote the communities detected by the algorithm. Only the vertex 3 (the yellow or light gray vertex) is incorrectly grouped to the red (dark gray) community. Indeed, it is a member of the blue (gray) community, as defined by the real problem. In the literature, the vertices 3 (e.g., see (Girvan and Newman, 2002)) and 10 (e.g., see (Newman, 2004)) are often misclassified by many community detection algorithms. This happens because the number of edges that they share between the two communities is the same, i.e., they are inherently overlapping, making their clustering a hard problem. In our technique, the overlapping vertex 10 is correctly classified.

3.4.4 Simulations for Data Clustering on Artificial and Real-World Data Sets

In order to measure the performance of the proposed method, we have conducted clustering experiments on both artificial and real-world data sets. Regarding the ex-

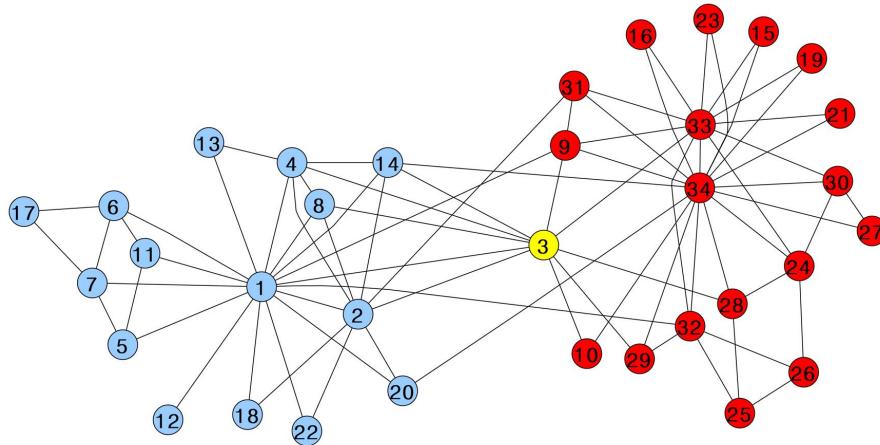


Figure 3.20: Community detection result of the Zachary's karate club network by using the proposed method. The red (dark gray) and blue (gray) colors denote the detected communities. Only the yellow or light gray vertex (vertex 3 in the original database) is incorrectly grouped.

periments on artificial data, we make use of some toy data sets with different data distributions, which are automatically generated via PRTools (Duin, 2000). For all simulations taken on artificial data, we have arbitrarily chosen the k -nearest neighbor graph formation technique with $k = 7$, i.e., each data item is represented by a vertex and each vertex is connected to its 7 nearest neighbors determined by the Euclidean distance among the data items. The initial locations of the particles are randomly chosen from a uniform distribution. The first data set consists of 300 samples equally divided into two banana-shaped classes. The result obtained by the Optimized K-Means (Khan, 2004) is exhibited in Fig. 3.21a, whereas Fig. 3.21b shows the results obtained by the proposed technique. The second data set is composed of 300 samples equally divided into two Highleyman classes. The result achieved by the Optimized K-Means is depicted in Fig. 3.21c, while Fig. 3.21d indicates the outcome encountered by the proposed technique. The third data set comprises 600 samples equally separated into two Lithuanian classes. The result attained by the Optimized K-Means is portrayed in Fig. 3.21e, whereas Fig. 3.21f reveals the result acquired by the proposed technique. All results obtained by our algorithm seem to be consistent to the logical structures of the data sets, on the contrary of Optimized K-Means algorithm, which can only detect spherical regions. A comparison to the CHAMELEON (Karypis *et al.*, 1999) technique is also made for these same data sets. The numerical results for the three techniques are summarized in Table 3.9. It can be observed that our algorithm has obtained better cluster detection accuracies than the techniques under comparison in the majority of the cases.

Now, we apply the proposed model to solve data clustering tasks on real-world data sets ranging from handwritten digits (Lecun's Modified NIST - see ([LeCun *et al.*, 1998](#))) to several well-known data sets available from the UCI Machine Learning

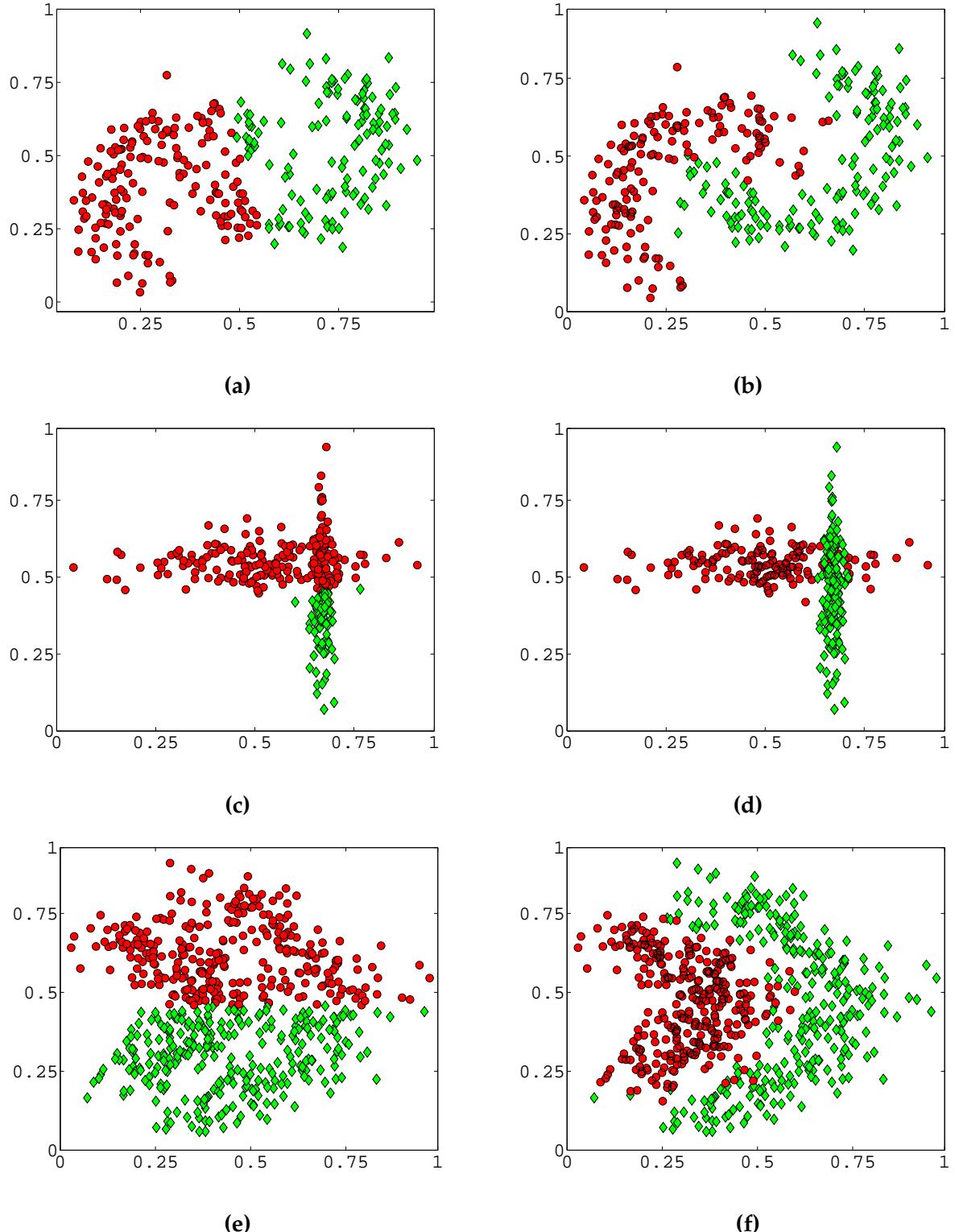


Figure 3.21: Data clustering of toy data sets with different cluster shapes. Data with the same color or format represent the same cluster. The left and right panels report the result obtained by the Optimized K-Means and the proposed technique, respectively. (a) and (b) banana-shaped clusters; (c) and (d) Highleyman clusters; (e) and (f) Lithuanian clusters.

Table 3.9: Data clustering accuracy for the distributions given in Fig. 5.13. The result is averaged over 20 runs.

	CHAMELEON	Optimized K-Means	Proposed Technique
Banana-shaped	97.00 ± 0.00	58.33 ± 6.39	97.67 ± 0.31
Highleyman	69.50 ± 0.00	61.73 ± 3.16	74.51 ± 2.03
Lithuanian	93.67 ± 0.00	43.48 ± 1.63	91.99 ± 2.92

Repository ([Frank and Asuncion, 2010](#)). Some metadata about the considered data sets are given in Table 3.10. For a detailed description, refer to ([LeCun et al., 1998](#)) for the Modified NIST (MNIST) data set, and ([Frank and Asuncion, 2010](#)) for the remaining data sets. Observe that, as we are interested in clustering matters (unsupervised learning), the available information about the class labels has been used only for evaluation of the results (data clustering accuracy).

Table 3.10: Databases' metadata description.

	# Instances	# Dimension	# Classes
Breast Cancer	699	9	2
Car Evaluation	1 728	6	4
Credit Approval	690	15	2
Contraceptive Method	1 473	9	3
Glass	214	9	6
Image Segmentation	2 310	20	7
Ionosphere	351	34	2
Iris	150	4	3
Vowel	90	10	11
Wine	178	13	3
Nursery	12 960	9	5
Letter Recognition	20 000	16	26
Modified NIST (MNIST)	10 000	784	10

Our algorithm is compared to a set of well-known techniques in optimal conditions, namely: the agglomerative hierarchical CHAMELEON algorithm ([Karypis, 2003](#)), Expectation Maximization algorithm with a Gaussian parametric family ([Dempster et al., 1977](#)), Fuzzy C-Means ([Bezdek, 1981](#)), K-Means with optimized center initialization ([Khan, 2004](#)), the adapted Modularity Greedy algorithm ([Clauset et al., 2004](#)), and Particle Swarm Optimization algorithm ([van der Merwe and Engelbrecht, 2003](#)). Among all these techniques, CHAMELEON and Modularity Greedy algorithm are deterministic algorithms, while the others are stochastic ones.

Regarding the proposed algorithm, the following configuration has been set for all simulations in this section: (i) all particles are randomly inserted into the network fol-

lowing a uniform distribution; and (ii) for all graph-based techniques, we have used the k -nearest neighbor graph formation technique with optimized k for each algorithm. The optimization is performed using the genetic algorithm available in the Global Optimization Toolbox of MATLAB with its default parameters. The parameters of the proposed algorithm are optimized over the following ranges: $0.5 \leq \lambda \leq 0.8$ and $1 \leq k \leq 30$. The other parameters are fixed according to Table 3.8. The optimization process consists of selecting the best combination of these two parameters which achieves the highest data clustering accuracy.

The simulation results are reported in Table 3.11. In this table, we have also provided the average rank of each algorithm, which is calculated as follows: (i) for each data set we rank the algorithms according to their average performance (average data clustering accuracy), i.e., the best algorithm is ranked as 1st, the second best one is ranked as 2nd, and so on; and (ii) for each algorithm, the average rank is given by the average value of its ranks scored on all the data sets. In order to examine the results in a statistical manner, we use an adaptation of (Demšar, 2006) for clustering tasks. The methodology described therein assigns a rank to each algorithm on each data set according to its final accuracy reached. After this step, the average rank of each algorithm is then evaluated and a Friedman Test is applied to the resulting average rank values of each algorithm. The Friedman Test is used to check whether the measured ranks are significantly distinct from the mean value of the ranks. In this case, the mean value of the ranks is 4, because there are 7 algorithms. The null-hypothesis considered here is that all the algorithms are equivalent, so their ranks should be the same. For all the future tests, we fix a significance level of 0.05. For our experiments, according to (Demšar, 2006), we have that $N = 13$ and $k = 7$, resulting in a critical value given by $F(6, 72) \approx 2.23$, where the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. In our case, we get a value $F_F \approx 8.05$ that is higher than the critical value, so the null-hypothesis is rejected at a 5% significance level.

As the null hypothesis is rejected, we are able to advance to post hoc tests which aim at verifying the performance of our algorithm in relation to others. For this task, we opt to use the Bonferroni-Dunn Test, with the proposed technique fixed as the control algorithm. According to (Demšar, 2006), one should not make pairwise comparisons when we test whether a specific method is better than others. Basically, the Bonferroni-Dunn Test quantifies whether the performance between an arbitrary algorithm and the reference is significantly different. This is done by verifying whether the corresponding average ranks of these algorithms differ by at least a critical difference (CD). If they do differ that much, then it is said that the better ranked algorithm is statistically superior to the worse ranked one. Otherwise, they do not present a significant difference for the problem at hands. Thus, if we perform the evaluation of the CD for

Table 3.11: Comparison between the proposed technique and several well-known techniques regarding the data clustering accuracy on real-world data sets. Each entry of the table presents the mean and its associated standard deviation averaged over 20 independent runs.

	CHAMELEON	Expectation Maximization	Fuzzy C-Means	Optimized K-Means	Modularity	Particle Swarm	Proposed Technique
Breast Cancer	95.57 ± 0.00	91.93 ± 0.49	93.92 ± 0.31	95.86 ± 0.46	95.85 ± 0.00	94.89 ± 1.32	95.90 ± 0.74
Car Evaluation	79.93 ± 0.00	71.03 ± 2.12	72.08 ± 1.92	72.92 ± 2.35	78.46 ± 0.00	75.09 ± 3.18	74.38 ± 1.43
Credit Approval	63.49 ± 0.00	62.94 ± 0.74	60.01 ± 0.26	58.34 ± 0.39	59.23 ± 0.00	60.39 ± 1.04	63.37 ± 0.39
Contraceptive Method	55.39 ± 0.00	44.25 ± 4.06	52.75 ± 2.95	54.94 ± 4.39	57.40 ± 0.00	50.07 ± 9.46	50.04 ± 4.50
Glass	53.73 ± 0.00	50.81 ± 6.30	52.48 ± 1.70	51.16 ± 2.41	51.40 ± 0.00	54.41 ± 15.62	52.92 ± 6.56
Image Segmentation	64.09 ± 0.00	61.30 ± 1.77	57.45 ± 3.05	58.88 ± 2.50	65.08 ± 0.00	62.37 ± 1.52	67.73 ± 1.36
Ionosphere	70.94 ± 0.00	73.08 ± 3.19	71.82 ± 0.95	70.00 ± 0.82	74.07 ± 0.00	71.83 ± 1.04	75.01 ± 2.98
Iris	88.67 ± 0.00	87.66 ± 1.95	88.16 ± 5.84	84.36 ± 8.75	95.33 ± 0.00	87.47 ± 5.38	89.57 ± 2.27
Vowel	47.78 ± 0.00	46.94 ± 3.43	44.30 ± 2.96	45.69 ± 2.15	48.89 ± 0.00	44.65 ± 2.55	47.52 ± 1.45
Wine	70.22 ± 0.00	71.34 ± 0.71	70.77 ± 0.61	68.54 ± 0.71	70.79 ± 0.00	71.35 ± 0.41	74.49 ± 1.62
Nursery	42.74 ± 0.00	36.66 ± 1.14	38.86 ± 2.90	38.26 ± 3.01	46.57 ± 0.00	41.01 ± 3.62	45.41 ± 2.29
Letter Recognition	91.58 ± 0.00	88.49 ± 2.89	88.19 ± 2.02	87.94 ± 2.04	86.11 ± 0.00	92.83 ± 2.93	91.37 ± 3.79
Modified NIST (MNIST)	75.33 ± 0.00	64.06 ± 3.11	61.21 ± 3.19	53.10 ± 3.38	70.94 ± 0.00	70.28 ± 3.19	74.53 ± 1.16
Average Rank	2.77	5.15	5.31	5.69	2.92	3.77	2.38

our problem, we encounter $CD = 2.24$. The average rank of the proposed method is 2.38. By virtue of that, if any rank does lie in the interval 2.38 ± 2.24 , the control algorithm and the compared algorithms are statistically equivalent. We conclude that our algorithm is superior to Expectation Maximization, Fuzzy C-Means, and Optimized K-Means under the terms previously stated. However, the other pairwise comparisons to the control algorithm do not surpass the CD, meaning that the differences among them are statistically insignificant. Therefore, the proposed algorithm presents no significant difference to CHAMELEON, Particle Swarm Optimization and Modularity Greedy Optimization algorithms for the given significance level. Nonetheless, the proposed technique has obtained the best performance (the best average rank) in relation to the other techniques.

Through the analysis of the shortcomings and advantages of the competing techniques and the proposed technique, we find that CHAMELEON is the only technique which realizes its clustering task in two steps: at the first step, it splits the network constructed from the original data set into very small subnetworks; at the second step, it merges these small sub-networks into larger ones, producing data clusters. The first step can be understood as a preparation process where well-defined small groups are formed. Once this is done, the possibility that these well-defined groups are wrongly split at the merging step is largely reduced. Such a preparation step is especially useful when the input data set is large or the groups are highly mixed. Thus, CHAMELEON is suitable for large data set clustering. However, such a processing introduces additional computational time. Moreover, CHAMELEON cannot handle outliers, since its prediction is based on two internal indices (closeness and interconnectivity). In this scenario, these indices would be seriously flawed. Another shortcoming of CHAMELEON is its termination criterion, which generally requires some domain knowledge. This feature, in turn, is commonly unknown *a priori*. This limitation is present because CHAMELEON is an agglomerative hierarchical algorithm and it is hard to know when to stop merging clusters or communities. If one wants to get good clustering results, high-processing efficiency, and, at the same time, wants to know the reasonable number of clusters presented in the data set, the technique proposed in this work would be a good choice.

3.4.5 Simulations for Detecting Overlapping Vertices and Communities

In this section, we give a number of examples with the purpose of assessing the effectiveness of the proposed technique with regard to detecting overlapping vertices. The ground truth (true group of each data item) of every used networked data set can be viewed by referring to its original papers that are cited here. By comparing the

results obtained by the proposed method and the results obtained by classical overlap vertex measures ([Fortunato, 2010](#); [Fu and Wang, 2003](#)), we see that the proposed technique is, in general, consistent to the classic vertex overlapping measures. In some situations, the proposed measure can get better results.

Zachary's Karate Club Network

First, we apply our technique to the Zachary's "karate club" network ([Zachary, 1977](#)), database in which we have already conducted some experiments in the previous section. As we have seen, only vertex 3 was incorrectly grouped. In the literature, the vertices 3 (e.g., see ([Girvan and Newman, 2002](#))) and 10 (e.g., see ([Newman, 2004](#))) are often misclassified by many community detection algorithms. This happens because the number of edges that they share between the two communities are the same; i.e., they are inherently overlapping, making their clustering a hard problem. In our technique, the overlapping vertex 10 is correctly classified. Now, we apply our overlapping index, as indicated in (3.27), on every vertex of the Zachary's "karate club" network. The result is shown in Fig. 3.22. One can see that the highest overlapping indices are exactly yielded by the vertices 3 and 10, which confirm our previous analysis. Moreover, the vertices 9, 14, 20, 29, and 32 also present a high level of overlapping characteristics.

Next, we will show that our model's performance does not depend on the initial locations of the particles. First, we put two particles into the vertex 17. Then, we put one at the vertex 17 and another at the vertex 27. The obtained results are very similar to the ones obtained in Fig. 3.22. In our model, the results do not depend on the order of presentation of data items due to the following factors:

1. In traditional neural networks, such as SOM, the data items are presented to the neural network one by one. In view of that, the order of presentation is very important ([Wang, 1997](#)). However, in our model, the whole data set is transformed into a graph at the same time, i.e., the constructed graph contains all the data items of a given data set. In this way, the clustering or the overlapping detection process do not depend on the data presentation order;
2. Due to the combined nature of random and preferential walking and the reanimation procedure of particles, the proposed model also does not depend on the initial positions of the particles. This point is confirmed by many computer simulations that here we describe. One example to show this feature is given by Fig. 3.22, where the same result is obtained from two quite different initial conditions.

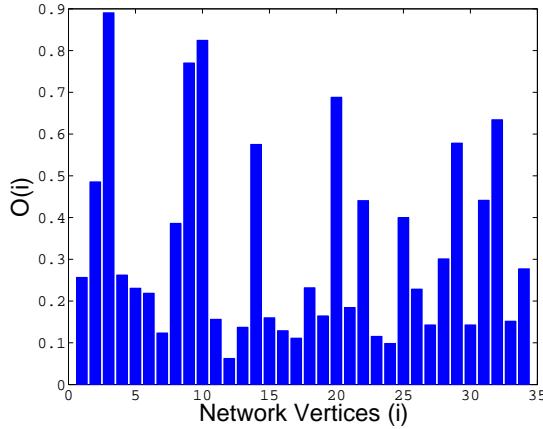


Figure 3.22: Result of the calculation of the overlapping index for all vertices in the Zachary's "karate club" network.

Dolphin Social Network

Continuing our analysis on overlapping vertices in complex network, in the next, we investigate a non-human social network named Dolphin Social Network ([Lusseau, 2003](#)), which is composed of 62 bottlenose dolphins living in Doubtful Sound, New Zealand. In this case, we consider that the dolphins are the vertices, whereas edges between dolphin pairs are established by observation when there is a statistically significant frequent association between them. Figure 3.23 indicates the community detection outcome of the proposed technique, along with the 5 most overlapping vertices depicted in larger sizes. In this case, the number of particles that maximizes $\langle R(t) \rangle$ is $K = 2$, which corresponds to the division of the real problem indicated by Lusseau. The communities are identified by the vertices' colors, in this case, blue (gray) and red (dark gray). The split into two groups seems to match the known division of the dolphin community, except for the dolphin "PL," which is a member of the blue (gray) community. Lusseau reports that, for a period of about two years during observation of the dolphins' behavior, the dolphins segregated into two communities, apparently by virtue of the disappearance of the dolphins located on the boundary between the communities. When some of these dolphins later reappeared, the two halves of the network joined together once more. Surprisingly, these border dolphins are the ones that the algorithm captured as being the 5 most overlapping vertices, as we can verify in Fig. 3.23 from the larger vertices, i.e., "DN63," "Knit," "PL," "SN89," and "SN100." As Lusseau draws attention to, developments of this kind illustrate that the dolphin network is not merely a scientific curiosity but, like human social networks, is closely tied to the evolution of the community. Nonetheless, we see that the complex networks are able to describe these types of problems in a natural and concise manner.

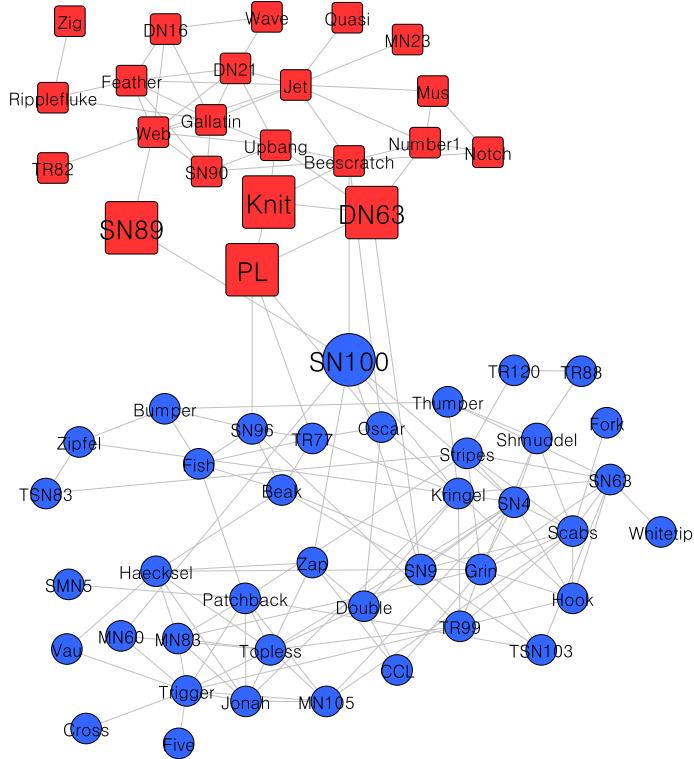


Figure 3.23: Dolphin Social Network observed by Lusseau. $K = 2$ and $\lambda = 0.6$. The 5 vertices with the highest overlapping structure are depicted with larger sizes.

Les Misérables Novel Network

Now we study the community structure of the interactions network between major characters comprising the Victor Hugo's sprawling novel of crime and redemption in post restoration France, denominated *Les Misérables*. Using the list of 77 character appearances by scene, compiled by Knuth (Knuth, 1993), the network was constructed in a way that vertices represent characters and an edge between two vertices represents co-appearance of the corresponding characters in one or more scenes (Newman, 2006a). In this case, the quantity $\langle R(t) \rangle$ is maximized when $K = 6$. Figure 3.24 shows the outcome of our technique, along with the 10 most overlapping vertices portrayed in larger sizes. The communities clearly reflect the subplot structure of the book. As one can expect, the protagonist Jean Valjean and his nemesis, the police officer Javert, are captured as being the 2 most overlapping vertices of the network, since they are central to the Hugo's play and form the hubs of communities composed of their respective adherents. The other detected overlapping vertices are the characters that

co-appear in multiple scenes, each of which with a different group of characters, as we can verify from the structure of the book.

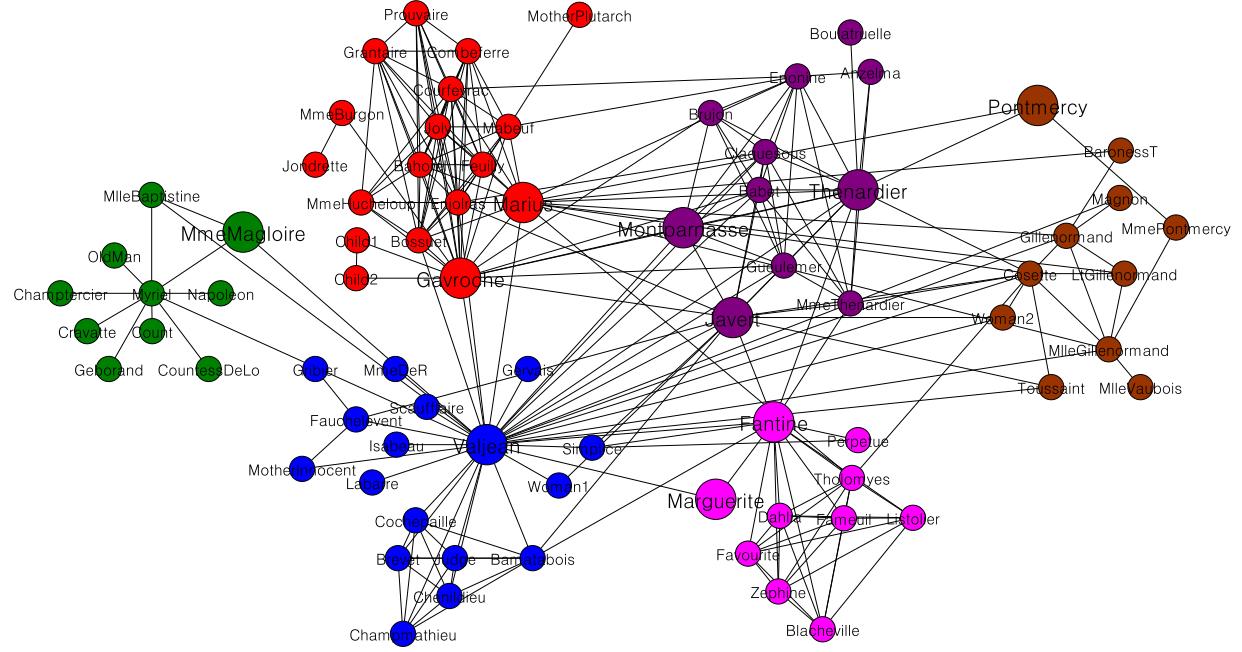


Figure 3.24: Hugo's sprawling novel of crime and redemption in post restoration entitled *Les Misérables*. $K = 6$ and $\lambda = 0.6$. The 10 vertices with the highest overlapping structure are depicted with larger sizes.

American College Football

In order to further verify the effectiveness of our overlapping index measure, we have opted to use another well-known data set named US College American Football (Girvan and Newman, 2002). In this case, the vertices in the graph represent each of the 117 teams and edges represent regular season games between the two teams they connect. Games are more frequent between members of the same conference than between members of different conferences; therefore, we expect that a structure with the presence of communities to be established. Figure 3.25 portrays the outcome of the algorithm, where the 15 vertices with the highest overlapping index are depicted with larger sizes. One can note that the majority of these overlapping vertices are also commonly misclassified by community detection techniques (see, for instance, (Girvan and Newman, 2002)), since they naturally belong to more than one community; i.e., these teams normally compete with rival teams from different conferences.

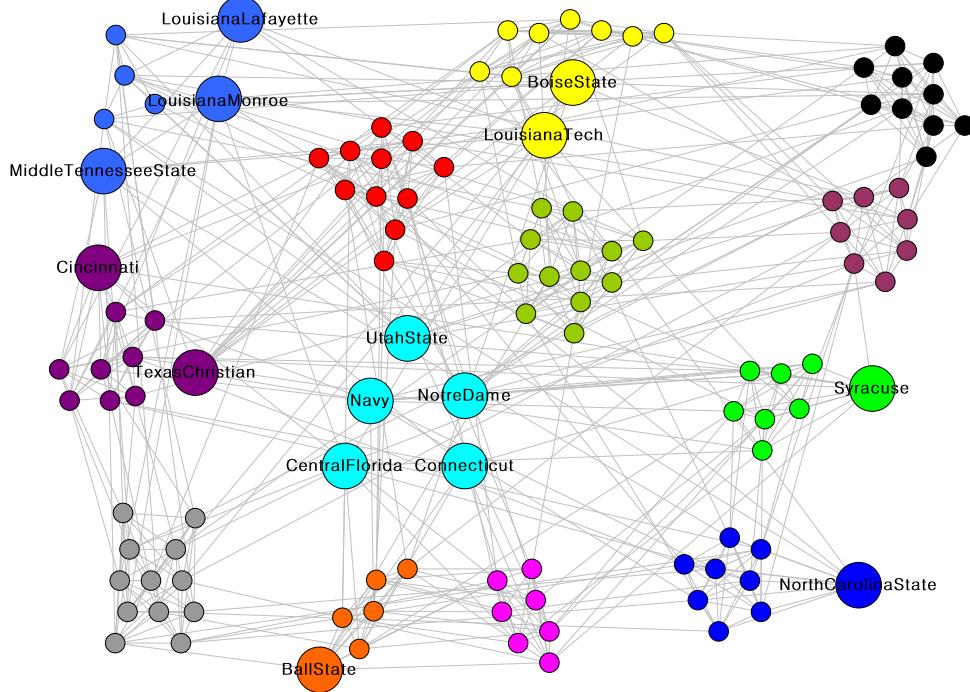


Figure 3.25: American College Football network. $K = 13$ and $\lambda = 0.6$. The 15 vertices with the highest overlapping structure are depicted with larger sizes.

Scientific Collaboration Network

For our next representative network, we assess the performance of the proposed technique on a collaboration network of scientists (Newman, 2006a). This network was constructed by gathering the names of authors which are present in the lengthy bibliography of Ref. (Newman, 2003b) and cross-referencing with the Physics E-print Archive at arxiv.org, specifically the condensed matter section of the archive where, for historical reasons, most papers on networks have appeared. Each of the 379 authors, which are seen in both repositories, are the vertices of the network, and an existing edge between two authors indicates coauthorship of one or more papers appearing in the archive. It is worth observing that the collaboration on any topic by a group of authors is a reasonable indicator of acquaintance; thus, this can be considered, to some extent, a social relationship network. Figure 3.26 reveals the outcome of the proposed technique applied only to the largest component of the network of collaborations between physicists who conduct research on networks. Comparing to the result given in (Newman, 2006a), we can verify that the algorithm can satisfactorily detect the communities in the underlying network. Furthermore, the 10 vertices with the largest overlapping index are depicted in larger sizes. One can see that these vertices, in the majority of times, are the authors that have coauthored with other physicists of different communities detected by the algorithm in a frequent manner. For instance, among the 10 most overlapping vertices, we can inspect that “Fortunato, S.” coauthored with two physi-

cists of his own detected community and one from another community; "Frauenfelder, H." coauthored with one of his community and one of another community. In brief, these 10 vertices share the common feature that they naturally belong to more than one community, since they receive a reasonable quantity of edges from other communities.

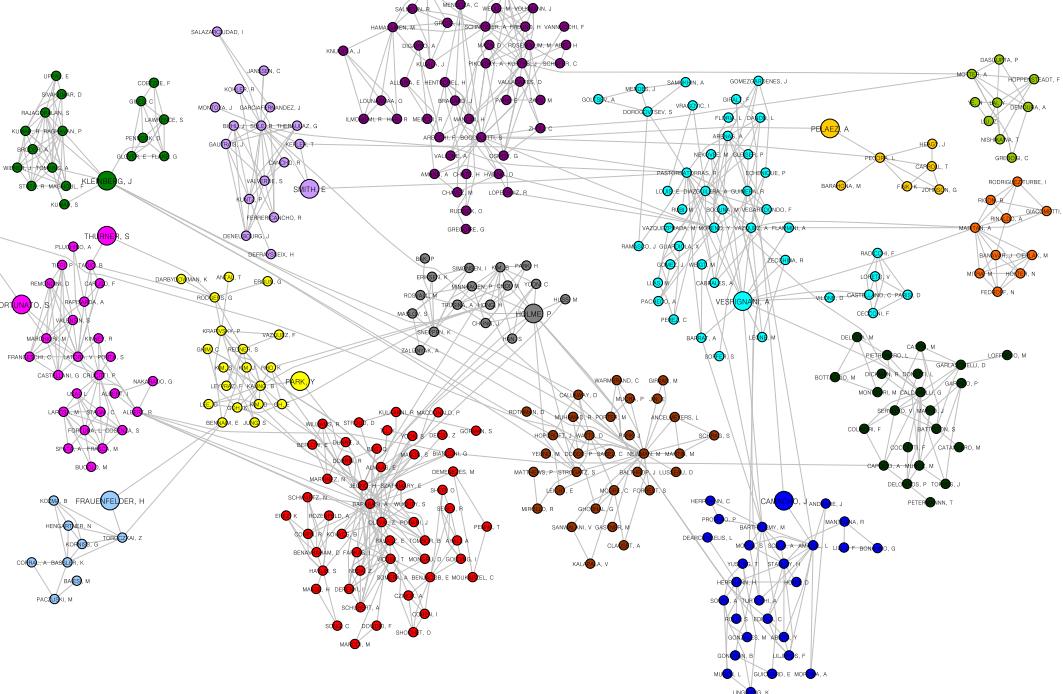


Figure 3.26: The Scientific Collaboration Network data set. Only the largest component is depicted. $K = 16$ and $\lambda = 0.6$. The 10 vertices with the highest overlapping structure are depicted with larger sizes.

3.5 Application: Handwritten Digits and Letters Clustering

In this section, we provide an application of data clustering for the competitive method for three real-world data sets based on handwritten digits and letters, which are the USPS, the MNIST, and the Letter Recognition data sets.

3.5.1 Motivation

Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-

screens, data sets, and other devices (Liu *et al.*, 2002; Theodoridis and Koutroumbas, 2008). Ideally, the handwriting recognition systems should be able to read and understand any handwriting (Bishop, 2007). Handwriting recognition has been one of the most fascinating and challenging research areas in the field of image processing and pattern recognition in the recent years (Pradeep *et al.*, 2011). It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications (Bishop, 2007; Theodoridis and Koutroumbas, 2008). In general, handwritten recognition is classified into off-line or on-line. In the first case, the writing is obtained by an electronic device and the captured writing is completely available as an image to the handwritten recognition method. In the second case, the coordinates of successive points are available by means of a function dependent on time, i.e., the complete image is not given (Pradeep *et al.*, 2011; Theodoridis and Koutroumbas, 2008). In summary, several research works have been proposed (Govindan and Shivaprasad, 1990; Mori *et al.*, 1992; Pradeep *et al.*, 2011) in an attempt to reduce the processing time of both off-line and on-line methods, while, at the same time, providing higher recognition accuracy. Due to the high complexity that this topic offers, it still has a wide range of problems to be addressed, such as the efficient recognition of images that are distorted or suffered a nonlinear transformation (Bishop, 2007; Duda *et al.*, 2001; Theodoridis and Koutroumbas, 2008). In view of these complexities, we will attempt to utilize complex networks to help in the task of handwritten digits and letters recognition by taking advantage of the topological characteristics of the constructed network of patterns in a competitive way.

3.5.2 The Network Formation Technique

In a graph-based data representation, the images (data items) are represented by the vertices, while the relationships between them are given by the links. A link connecting two vertices (images) holds a weight that numerically translates the similarity between them. Each image can be represented by a "square" matrix $\eta \times \eta$. For rectangle images, a pre-processing is required to transform it into a square image. We conventionally set the pixels' values range to lie within the interval $[0, 1]$ by normalization. Thus, an arbitrary data item (image) x_i can be seen as a matrix with dimensions $\eta \times \eta$, where each pixel $x_i^{(u,j)} \in [0, 1], \forall (u, j) \in \{1, \dots, \eta\} \times \{1, \dots, \eta\}$.

In order to construct the network, we are required to establish a similarity measure. The traditional pixel-per-pixel distance is rather insufficient in terms of reliably representing data, since such measure is very sensitive to rotations and scale modifications. With the purpose of overcoming this difficulty, we propose a measure based on the eigenvalues that each image inherently carries with it. First of all, we remove the mean associated to each data item (image), so that we have a common basis of comparison.

After that, we calculate the ϕ greatest eigenvalues of the image. Efficient methods have been developed for finding the leading eigenvalues of real-valued asymmetric matrices (Goldhirsch *et al.*, 1987; Tsai *et al.*, 2010). The magnitudes of the eigenvalues are related to the variations that the image possesses; hence, it is a natural carrier of information (Jolliffe, 2002). The greater its value, more information about the image it conveys. By virtue of that, a good choice is to only extract the greatest $\phi < \eta$ eigenvalues and drop the smaller values, since these do not transport too much information about the image. Also, in order to give more emphasis to the largest eigenvalues, a weight is associated to each one so that the larger an eigenvalue is, the larger will be its associated weight.

Consider that we are to compare the similarity between two images, say x_i and x_j , in relation to the ϕ largest eigenvalues. We firstly sort the ϕ eigenvalues of each image as: $|\lambda_i^{(1)}| \geq |\lambda_i^{(2)}| \geq \dots \geq |\lambda_i^{(\phi)}|$ and $|\lambda_j^{(1)}| \geq |\lambda_j^{(2)}| \geq \dots \geq |\lambda_j^{(\phi)}|$, where $|\lambda_i^{(k)}|$ marks the k th eigenvalue of the i th data item. In this case, the dissimilarity ρ (or, equivalently, the similarity $1 - \rho$) between image i and j is given by:

$$\rho(i, j) = \frac{1}{\rho_{\max}} \sum_{k=1}^{\phi} \beta(k) \left[|\lambda_i^{(k)}| - |\lambda_j^{(k)}| \right]^2 \quad (3.84)$$

where $\rho \in [0, 1]$, $\rho_{\max} > 0$ is a normalization constant, $\beta : \mathbb{N}^* \rightarrow (0, \infty)$ indicates a monotonically decreasing function that can be arbitrary chosen by the user.

3.5.3 Brief Information of the Handwritten Digits and Letters Data Sets

The data sets in which the proposed model will be tested against are given in the following:

- *USPS Data Set*: Comprised of 9298 images of handwritten digits. The digits 0 to 9 have 1553, 1269, 929, 824, 852, 716, 834, 792, 708, and 821 samples respectively. The USPS digits data were gathered at the Center of Excellence in Document Analysis and Recognition (CEDAR) at SUNY Buffalo, as part of a project sponsored by the US postal Service. For more details about this data set, refer to (Hull, 1994). Each image has dimensions of 16×16 pixels, with 256 grey levels per pixel. We will employ the weighted eigenvalue similarity measure as studied before. Instead of using 16 eigenvalues, we will only work with the 4 greatest ones. In this case, the β function, as shown in (3.84), will be fixed as an exponential decreasing function with a time constant fixed at $\tau = 3$ and a scaling factor given by 16, i.e., $\beta(x) = 16 \exp(\frac{x}{3})$. Since, this function is mapped into the interval $(0, \infty)$ and is a monotonically decreasing function, it follows that this

β function meets all the aforementioned requirements. Specifically in this situation, we have that the weights associated to each eigenvalue are: $\beta(1) = 11.46$, $\beta(2) = 8.21$, $\beta(3) = 5.89$, and $\beta(4) = 4.22$.

- *MNIST Data Set:* Originally composed of images with dimensions 28×28 . We will only use the public set composed of 10 000 vertices. Moreover, we will make use of the dissimilarity measure based on the first 4 eigenvalues of each image out of 28 eigenvalues. The same β function employed in the USPS data set will be used here.
- *Letter Recognition Data Set:* Composed of characteristic vectors with 16 entries. There are 20 000 vertices.

Since none of these data sets are in a network form, the methodology is divided into two general steps: the network formation and the data clustering tasks. In the first, we use the k-nearest neighbor network formation technique with $k = 3$ after we apply a pre-processing step. In this pre-processing, we standardize the data such as to have zero mean and unitary standard deviation. As for the distance measure, we either use the proposed weighted eigenvalue dissimilarity (for the first two data sets above) or the reciprocal of the Euclidian similarity (for the last one). The reason behind not using the weighted eigenvalue on the third data set is because the samples are not provided as images, but as image descriptors. Since the latter is formed by merely scalars, we cannot apply the proposed dissimilarity measure. In the second step, the data clustering algorithm based on particle competition is applied. As we are dealing with unsupervised learning, we do not use any external information, such as labels or exogenous knowledge. Instead, we limit ourselves to discovering explicit or implicit relationships among the data by the mechanism of particle competition.

3.5.4 Determining the Optimal Number of Particles and Clusters

Figures 3.27a, 3.27b, and 3.27c show the determination of the optimal K for the USPS, MNIST, and Letter Recognition data sets, respectively. One can verify that $\langle R(t) \rangle$ is maximized exactly when the number of particles is equal to the number of clusters in the network, confirming the effectiveness of such heuristic.

3.5.5 Handwritten Data Clustering

Here, we report the cluster detection accuracy reached by our algorithm in detail, along with the data clustering accuracy of a selected set of competing techniques. For the calculation of the cluster detection accuracy, we set that the ideal result is that each cluster represents a "digit" (in the USPS and MNIST data set) or a "letter" (in the Letter

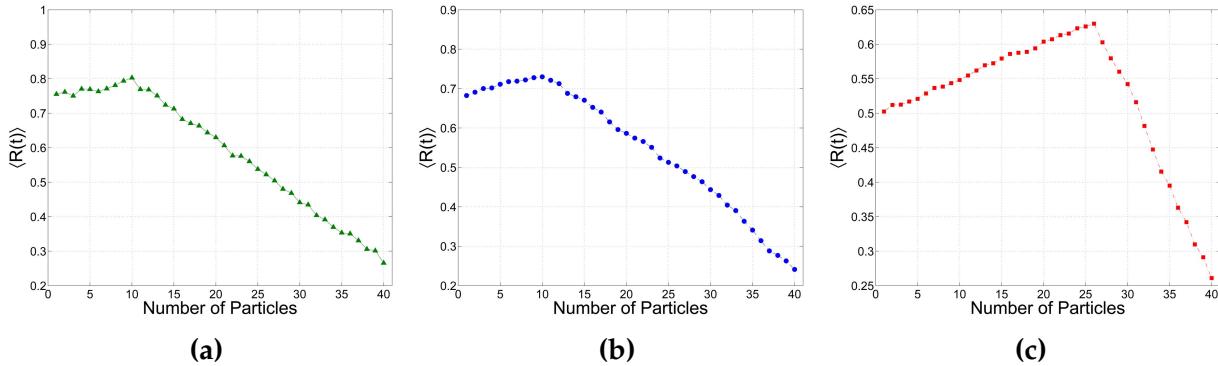


Figure 3.27: Determination of the optimal number of particles K (the optimal number of clusters) in real-world data sets. The number of classes that each data set originally possesses are: (a) The USPS data set has 10 clusters (each cluster corresponding to a number from “0” to “9”); (b) The MNIST data set has 10 clusters (each cluster corresponding to a number from “0” to “9”); and (c) The Letter Recognition data set has 26 clusters (each cluster corresponding to a letter from the English alphabet (“A” to “Z”)). 20 independent runs are performed and the average value is reported.

Recognition data set). Particularly, Table 3.12 supplies details about the algorithms that we have chosen for comparison matters. We have used the genetic algorithm available in the Global Optimization Toolbox of MATLAB with its default parameters in order to optimize the parameters of our algorithm. In our case, we have optimized λ over the range $0.2 \leq \lambda \leq 0.8$. This optimization process is conducted in a way to choose the best λ for which the data accuracy reaches its maximum value. By doing this, we get that the optimal values of λ for the USPS, MNIST, and Letter Recognition data sets to be 0.58, 0.60, 0.60, respectively. The number of particles to be inserted is determined accordingly to the previously analysis that we have shown, i.e., with the aid of the $\langle R(t) \rangle$ measure. We choose the number of particles that maximizes this quantity. By looking at Figs. 3.27a (USPS), 3.27b (MNIST), and 3.27c (Letter Recognition), we are able to conclude that the maximum value of $\langle R(t) \rangle$ is attained when the number of particles is 10, 10, and 26, respectively.

Table 3.12: Description of the competing state-of-art data clustering techniques.

Technique	Reference
Gaussian Mixture Model (GMM)	(Bishop, 2007)
K-Means	(MacQueen, 1967)
Locally Consistent Gaussian Mixture Model (LCGMM)	(Liu <i>et al.</i> , 2010)
Spectral clustering algorithm with normalized cut (Ncut)	(Shi and Malik, 1997)
Ncut Embedding All (NcutEmb ^{All})	(Ratle <i>et al.</i> , 2008)
Ncut Embedding Maximum (NcutEmb ^{Max})	(Ratle <i>et al.</i> , 2008)

Table 3.13 reports the data clustering accuracy reached by our method and the aforementioned competing algorithms. Some of these results are readily extracted

Table 3.13: Data clustering accuracy reached by the proposed technique and the competing methods listed in Table 3.12. For the stochastic methods, such as the particle competition method, thirty independent runs were performed and the corresponding mean is provided.

	USPS	MNIST	Letter Recognition	Avg. Rank
LCGMM	73.83	73.60	93.03	2.33
GMM	67.30	66.60	91.24	5.33
K-Means	69.80	53.10	87.94	6.33
NCut	69.34	68.80	88.72	5.67
NCutEmb^{All}	72.72	75.10	90.07	3.67
NCutEmb^{Max}	72.97	75.63	90.59	2.67
Proposed Technique	80.46	74.53	91.37	2.00

from (Ratle *et al.*, 2008) and (Liu *et al.*, 2010). For more information about the parameters used in the competing techniques, see the aforementioned references. Within this table, we have provided the Average Rank of each algorithm, which is calculated as follows: (i) for each data set we rank the algorithms according to their average performance (average data clustering accuracy), i.e., the best algorithm is ranked as 1, the second best one is ranked as 2, and so on; (ii) for each algorithm, the Average Rank is given by the average value of its rank achieved in all the data sets. As we can verify by looking at the Average Rank column, our algorithm has reached one of the best positions, showing the effectiveness of the proposed technique. In order to examine the results in a statistical manner, we utilize an adaptation of (Demšar, 2006) for clustering matters. The methodology described therein assigns a rank to each algorithm on each data set according to its final accuracy reached. After this step, the average rank of each algorithm is then evaluated and a Friedman Test is applied to the resulting average rank values of each algorithm. The Friedman Test is used to check whether the measured average ranks are significantly distinct from the mean rank, in this case 4.0, because there are seven data clustering techniques. The null-hypothesis considered here is that all the algorithms are equivalent, so their ranks should be equal. Hereon, for all the future tests, we fix a significance level of 0.10. For our experiments, according to (Demšar, 2006), we have that $N = 3$ and $k = 7$, resulting in a critical value given by $F(6, 12) = 2.33$, where the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. In our case, we get a value $F_F \approx 4.00$ that is higher than the critical value, so the null-hypothesis is rejected at a 10% significance level. Thus, we conclude that the algorithms at hand present statistical difference, i.e., they are statistically different from the mean rank. Nonetheless, one can see that our algorithm has obtained the best average rank in relation to the other algorithms for these three data sets.

As the null hypothesis is rejected, we are able to advance to post hoc tests which aim

at verifying the performance of our algorithm in relation to others. For this task, we opt to use the Bonferroni-Dunn Test, with the proposed technique fixed as the control algorithm. According to (Demšar, 2006), one should not make pairwise comparisons when we test whether a specific method is better than others. Basically, the Bonferroni-Dunn Test quantifies whether the performance between an arbitrary algorithm and the reference is significantly different. This is done by verifying whether the corresponding average ranks of these algorithms differ by at least a critical difference (CD). If they do differ that much, then it is said that the better ranked algorithm is statistically superior to the worse ranked one. Otherwise, they do not present a significant difference for the problem at hands. Thus, if we perform the evaluation of the CD for our problem, we encounter $CD = 4.22$ when the significance level is 10%. The average rank of the proposed method is 2.00. By virtue of that, if any rank does lie in the interval 2.00 ± 4.22 , the control algorithm and the compared algorithms are statistically equivalent. We conclude that our algorithm is superior to K-Means for this set of databases. However, the other pairwise comparisons to the control algorithm do not surpass the CD, meaning that the differences among them are statistically insignificant. Nonetheless, the proposed technique has obtained the best performance (the best average rank) in relation to the other techniques and also presents reasonable computational complexity as we have previously addressed.

In order to further verify the robustness of the proposed technique, we inspect the samples that compose a same cluster. Specifically, Figs. 3.28, 3.29, 3.30, and 3.31 show some samples of the clusters representing the pattern “2”, “5”, “6”, and “8”, respectively, of the MNIST data set. These samples are captured using the following strategy: we compute the vertices that compose the maximum geodesic distance of the cluster representing each pattern (cluster diameter). Now, we select a representative subset of vertices composing the cluster diameter trajectory for illustrative purposes. In these figures, samples that are adjacent are more similar than those distant from one to another. On the basis of this analysis, we conclude that the graph representation has successfully captured several variations of the these number patterns each of which in a single representative cluster, showing the robustness of the proposed model.

Lastly, we intend to study the overlapping characteristics of the MNIST data set. In order to do so, we report in Table 3.14 seven of the most overlapping vertices encountered by the proposed technique. For each sample, we provide the domination level imposed by each particle on that vertex. Clearly, we can see that the majority of the samples are frequently visited by more than 1 particle in the network; hence, its overlapping trait. These vertices are commonly misclassified by the majority of the clustering techniques (LeCun *et al.*, 1998). For instance, the first row shows a “0” sample that is often visited by particles 0 and 6, each of which imposing 0.37 and 0.40 of domination level, respectively. One can see that the samples in the rows 2, 3, 4, and 6

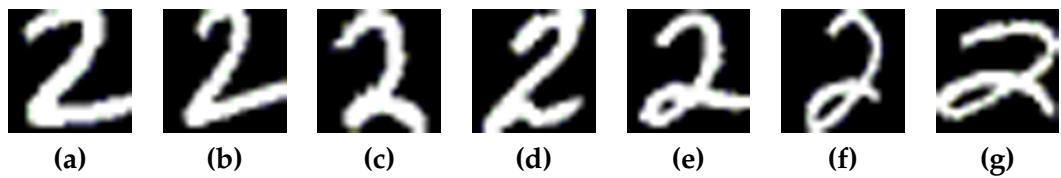


Figure 3.28: A broad set of samples of that were classified as being member of the cluster representing the pattern “2”. Note that samples that are adjacent are similar with regards to the weighted eigenvalue dissimilarity function. The transitions from the sample (a) to (g) were captured from the maximum geodesic distance between two vertices in the cluster representing pattern 2. In this case, the diameter of such cluster is 17. We have only provided 7 representative samples above.

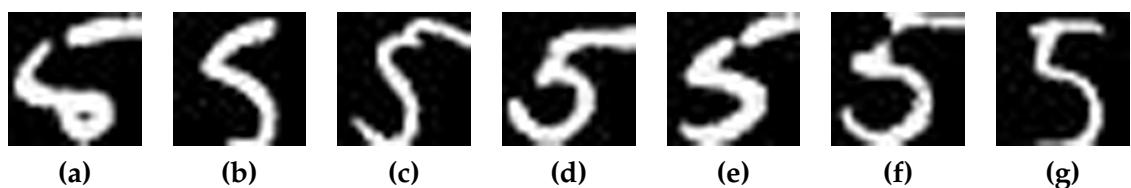


Figure 3.29: A broad set of samples of that were classified as being member of the cluster representing the pattern “5”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples.

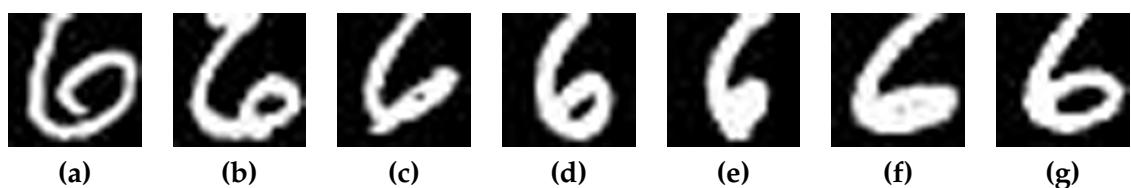


Figure 3.30: A broad set of samples of that were classified as being member of the cluster representing the pattern “6”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples.

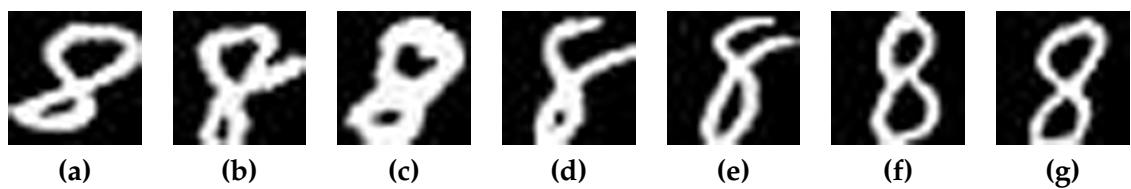


Figure 3.31: A broad set of samples of that were classified as being member of the cluster representing the pattern “8”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples.

are correctly classified, in spite of having a big uncertainty revolving around its true group, since the overlapping index of these samples occurred to be 0.94, 0.96, 0.89, and 0.97, respectively. Therefore, these overlapping indices are on the verge of reaching the maximum possible value, in the case, 1, empirically confirming its high overlapping

characteristics.

3.6 Chapter Remarks

The work explored in this chapter, which has been published in the *IEEE Transactions on Neural Networks and Learning Systems* (Silva and Zhao, 2012a), has received featured comments as “CIS Publication Spotlight” in the *IEEE Computational Intelligence Magazine*, page 12, August 2012 (Liu *et al.*, 2012). The application of the same algorithm for handwritten digits and letters clustering published in SIBGRAPI 2011 (Silva *et al.*, 2011a) has rendered the award of one of the Best Papers of the referred conference (SIBGRAPI, 2011).

Table 3.14: List of some samples of the MNIST data set that are classified as possessing high overlapping characteristics. For every sample, the domination level imposed by all $K = 10$ particles on that sample is reported. The algorithm groups or classifies a sample according to particle that enforced the highest domination level (in bold). 10 independent runs were performed and the mean value is informed.

Sample	Domination Levels of each particle										Real Class
	0	1	2	3	4	5	6	7	8	9	
	0.37	0.02	0.01	0.01	0.01	0.02	0.40	0.01	0.12	0.03	“0”
	0.39	0.01	0.02	0.02	0.02	0.07	0.03	0.01	0.33	0.10	“0”
	0.04	0.19	0.31	0.01	0.02	0.02	0.03	0.27	0.03	0.08	“2”
	0.01	0.03	0.01	0.45	0.01	0.02	0.01	0.34	0.10	0.02	“3”
	0.02	0.05	0.02	0.01	0.31	0.10	0.02	0.02	0.09	0.36	“4”
	0.02	0.01	0.01	0.01	0.03	0.38	0.35	0.01	0.16	0.02	“5”
	0.01	0.01	0.03	0.05	0.44	0.06	0.02	0.05	0.07	0.26	“9”

In this chapter, a rigorous definition of a new model for competitive learning in complex networks has been proposed, whose foundations were biologically inspired

by the competition process taking place in many nature and social systems. In this model, several particles navigate in the network to explore their territory and, at the same time, attempt to defend their territory from rival particles. If a particle frequently visits a specific vertex, it occurs that the domination level of the visiting particle on that vertex is strengthened; simultaneously, the domination levels of all the other particles on the same vertex are weakened. Finally, each particle is confined in a community of the network.

The proposed model is nonlinear and stochastic. Owing to the mathematical formality that the model was built upon, we have been able to conduct a theoretical and empirical analyses to better study the evolutional behavior of the model. A convergence analysis has shown that the model presents structural stability of the dynamical system rather than asymptotic stability, which is a welcomed characteristic, since it better describes the uncertainty that revolves around real-world problems, which have noise and uncontrolled variables. In addition, due to this analysis, we have found that the model is a generalization of the process of single independent random walkers in a network. Specifically, we have shown that the model's behavior acts as multiple interacting walkers in a network. The interaction is molded in a competitive way, by using a probabilistic convex combination of random and preferential walks. Such generalization is realized by calibrating the values of the parameter λ and Δ of the system. If $\lambda = 0$, the proposed model reduces to multiple non-interacting random walks; but, when $\lambda > 0$, the competitive mechanism is turned on.

Furthermore, we have deduced a new measure for detecting overlapping structures and for estimating the number of actual clusters or communities in a network, whose calculations are embedded into the model's own algorithm. This permits their calculation to be performed in an efficient way.

Simulations have been carried out with the purpose of quantifying the robustness of the proposed technique on artificial and real-world data sets for the tasks of data clustering and community detection. Computer simulations have revealed that the proposed model works well for community detection and for data clustering tasks. Finally, an application on handwritten digits and letters recognition has been provided and high clustering accuracies have been obtained. Moreover, we have analyzed the composition of the clusters formed in the MNIST data set and have verified that, within a specific cluster, several variations of the same pattern can be encountered, confirming the robustness of the model. More importantly, this work is an attempt to provide an alternative way to the study of competitive learning.



CHAPTER
4

Semisupervised Stochastic Competitive Learning in Complex Networks

This chapter treats the issue of semisupervised classification in complex networks by adapting and enhancing the particle competition algorithm that we have described in the previous chapter. Specifically, this enhancement will be achieved by introducing the idea of cooperation among the particles and by changing the inner mechanism of the original algorithm so as to fit it in a semisupervised environment. In contrast to the unsupervised learning model, where the particles are randomly spawned in the network because no prior analysis of the groups is available, the semisupervised learning version does have some external knowledge by definition. This knowledge is represented by the labeled data items, usually offered as a small fraction of the whole data set. In this scenario, the objective is to propagate the labels from the labeled set to the unlabeled set. Likewise the previous chapter, here we also provide a mathematical formalization of the model, as well as a theoretical analysis. A great portion of this analysis is based on the model that we have studied in the last chapter. A validation is also presented linking the numerical and theoretical results.

Once the basic concepts are properly presented, we will deal with the interesting problem of learning with imperfect data. In this case, the labeled data set is not totally reliable, because the external professor may incorrectly label data items. We will further enhance the model to be able to tackle this task by providing mechanisms to detect and to prevent wrongly labeled data. For this end, only the information generated by the competitive process itself is considered. Therefore, these mechanisms are embedded within the model. Afterwards, we show that the modified model can really provide good results even in environments where the reliability of the data is small. In-

vestigations and analyses of these introduced mechanisms are going to be presented, so as to one better understand the potentialities and shortcomings of the proposed model. Critical points, which are recognized by the points where the accuracies down-fall in a quick manner, reveal that the model can withstand a noisy environment in a satisfactory manner, provided that the fraction of correctly labeled vertices is the majority. This is intuitive and confirms that the competitive model, as time progresses, by a kind of “natural selection,” makes the majority overwhelms the minority using the topology of the network.

4.1 Model Description

In this section, we deal with the description of the semisupervised version of the particle competition model in detail.

4.1.1 A First Glimpse at the Differences of the Semisupervised and the Unsupervised Versions

As we have seen, in the machine learning theory, the semisupervised learning differs from the unsupervised learning by the fact that the former has some external knowledge incorporated into the beginning of the learning process, by means of pre-labeled data items. Having this concept at mind, the main difference of the semisupervised version of the method resides in the fact that each particle now represents a labeled data item, and its main goal is to spread the label of its represented vertex by visiting and dominating the neighborhood in a competitive way. Naturally, in this case, each particle is always a representative of a labeled vertex, which we denote as its home vertex. In the reanimation procedure of a particle, it no longer randomly chooses a dominated vertex to properly recharge its energy level; rather, it always regresses to its home vertex, which is always strongly dominated by it, in order to become active again. As we will see, the introduction of this modification leads the model to work in a local label-spreading fashion basis, since each particle is probabilistically bounded within a small region of the network. As a consequence, due to the competitive mechanism, each particle only visits a portion of vertices potentially belonging to the current particle, while it is not allowed to visit those vertices definitely occupied by other particles. This concept can be roughly conceived as a “divide-and-conquer” effect embedded into the competitive-cooperative scheme.

Revolving around this new model, it may occur that, if there are more than one labeled data item with the same label, then the corresponding representative particles will act together as a team. This happens because they are propagators of the same labels; hence, it is natural to think that they will join together and will work as a team.

In this way, several teams compete with each other to establish their class borders, while cooperating with its teammates.

4.1.2 Familiarizing with the Environment

In the semisupervised learning scheme, consider that $\mathcal{L} = \{L_1, L_2, \dots, L_L\}$ denotes the set of possible discrete classes that can be predicted by the semisupervised classifier. A set of data items $\mathcal{X} = \{x_1, \dots, x_l, x_{l+1}, \dots, x_{l+u}\}$ is supplied, where each entry is a d -dimensional vector in the form $x_i = \{x_{i1}, \dots, x_{id}\}$. The first l data items are initially labeled and compose the labeled set \mathcal{X}_l . For each $x_i \in \mathcal{X}_l$, a label $c_i \in \mathcal{L}$ is given. The remaining data items comprise the unlabeled data set \mathcal{X}_u and no labels are known for these items. The objective is to propagate the labels from \mathcal{X}_l to \mathcal{X}_u , while preserving the data distributions. In practice, the proportion of unlabeled data items far surpasses the proportion of labeled data items, such that $u \gg l$ is satisfied in several occasions.

Since the particle competition model is a network-based technique, a network formation technique must be employed to transform the vector-based data. For this end, the data are mapped into a graph \mathcal{G} using a network formation technique $g : \mathcal{X} \mapsto \mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \mathcal{V}_l \cup \mathcal{V}_u = \{1, \dots, V\}$ is the set of vertices and \mathcal{E} is the set of edges. In addition, \mathcal{V}_l and \mathcal{V}_u indicate the data items in \mathcal{X}_l and \mathcal{X}_u which have become vertices, respectively. Each vertex in \mathcal{V} represents a data item in \mathcal{X} . The edges in \mathcal{E} may be created using the ϵ -radius, the k -nearest neighbors (k -NN) graph formation techniques, or any techniques that may be convenient for the problem at hand. In the original versions, the ϵ -radius technique creates a link between two vertices if they are within a distance ϵ , while the k -NN sets up a link between vertices i and j if i is one of the k nearest neighbors of j or vice versa.

4.1.3 Deriving the Competitive Transition Matrix of the Dynamical System

In this section, we focus on the technical differences of the unsupervised and semisupervised learning transition matrices. If any part of the method has not been expressly indicated here, then it means that it is identical to the unsupervised transition matrix derived in Section 3.1.2.

The transition matrix takes exactly the same form as the unsupervised version, which, for convenience, we remember as follows:

$$\mathbb{P}_{\text{transition}}^{(k)}(t) \triangleq (1 - S^{(k)}(t)) \left[\lambda \mathbb{P}_{\text{pref}}^{(k)}(t) + (1 - \lambda) \mathbb{P}_{\text{rand}}^{(k)} \right] + S^{(k)}(t) \mathbb{P}_{\text{rean}}^{(k)}(t). \quad (4.1)$$

Basically, the technical differences are reflected on how each of these matrices com-

prising the transition matrix are defined. Specifically, the random and preferential terms do not suffer any modifications. However, the reanimation matrix must be adapted in order to the algorithm to work in a local label-spreading fashion basis, as well as to comport the idea of home vertex that we have previously described. First, it worth remembering that each entry of $\mathbb{P}_{\text{rean}}^{(k)}(t)$ is responsible for indicating the probability of teleporting an exhausted particle $k \in \mathcal{K}$ back to its dominated territory. Here, we always transport the particle back to its home vertex, which is the vertex that particle k represents. Suppose that particle k is visiting vertex i when its energy becomes completely depleted. In this particular occasion, this transportation will obey the following distribution:

$$\mathbb{P}_{\text{rean}}^{(k)}(i, j, t) \triangleq \begin{cases} 1, & \text{if } j = v_k \\ 0, & \text{otherwise} \end{cases}. \quad (4.2)$$

where v_k indicates the home vertex of particle k , which is essentially the vertex that particle k is representing in the competitive process. Therefore, matrix $\mathbb{P}_{\text{rean}}(t)$ will only have a non-zero entry for each particle at any given time t , which is the j -entry that is exactly the particle's home vertex v_k . In computational terms, this can greatly enhance the process of obtaining the next vertex that particle k will visit. For didactic purposes, Fig. 4.1 portrays a simple scenario of a reanimation taking place. In this case, the red or dark gray particle, since it is visiting a vertex dominated by a rival particle, will have its energy penalized. Supposing that its energy has been completely depleted, then the particle will become exhausted. Under these circumstances, the reanimation of such particle will occur, which will compel the particle to travel back to its home vertex to be properly recharged. Even though this is a simple mechanism, we have assessed that it can greatly increase the performance of the particle competition model, because it does not let particles go wander very far from their origins.

4.1.4 The Initial Conditions of the System

In order to run system ϕ , we need a set of initial conditions. Firstly, the particles' initial position vector $p(0)$ is user-controllable. The initial positions of the particles do not affect the classification process, due to the reanimation process. Usually, the particles are spawned in their home vertices. Secondly, in order to adapt such model to the semisupervised scheme, we perform a customized initial condition on the matrix $N(0)$. For those initially labeled vertices, we fix their owner to be the particle that is generated there in the following manner: as the ownership is represented by the maximum actual domination level imposed on that vertex, we simply force the number of visits of the representative particle to its home vertex to be infinity at the beginning; thus,

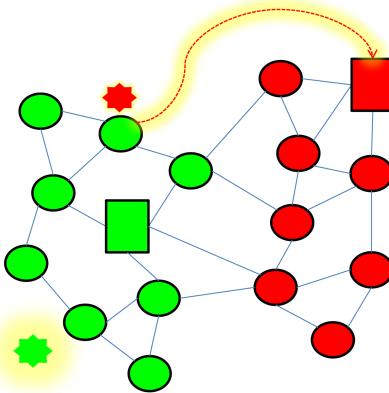


Figure 4.1: A reanimation schematic of an exhausted particle. The square-shaped vertices denote pre-labeled vertices, whereas the circle-shaped ones, unlabeled vertices. Two different classes are possible: the red or dark gray and the blue or gray classes. We fill in the vertex with the color of the particle which is imposing the highest domination level. The continuous edges represent the topology of the network, and the dotted line represents the only available path for the exhausted red or dark gray particle, which is always its home vertex, i.e., the red or dark gray squared-shaped vertex. The yellow or light gray glowing emanating from the particles symbolizes their current energy. As the particles gets more energized, the higher will be its glowing power in the drawing.

making impossible owner switching over that home vertex, which is what we want. Usually, more than one particle (a team) is generated to represent a set of pre-labeled examples of the same class. Each of them tries to dominate vertices independently. The cooperation among the particles of the same team happens only at the end of the process. In order to do so, for each vertex, we sum up the domination levels of all particles of the same team on it to obtain the aggregated domination level. In view of this scheme, we have that each entry of $N(0)$ is given by:

$$N_i^{(k)}(0) = \begin{cases} \infty, & \text{if particle } k \text{ represents vertex } i \\ 1 + \mathbb{1}_{\{p^{(k)}(0)=i\}}, & \text{otherwise} \end{cases}, \quad (4.3)$$

where we apply (4.3) to every $(i, k) \in \mathcal{V} \times \mathcal{K}$. Note that the scalar 1 was used in the second expression of (4.3) in order to unlabeled and not visited vertices at time 0 to have their calculation well-defined, according to (3.7).

4.1.5 Discovering the Computational Complexity

In the semisupervised learning version, we must always construct the network from the vectorized data. Such a task always runs in $\mathcal{O}(V^2)$ time complexity order. From the time complexity analysis that we have investigated in Section 3.1.6, we have seen that the proposed particle competition model runs roughly in linear time com-

plexity order. Since these two steps are performed in a serial manner, then the semisupervised particle competition algorithm always runs in $\mathcal{O}(V^2)$ time complexity order, because of the network formation step. However, if a networked data is supplied to the algorithm, then the semisupervised particle competition algorithm runs in linear time, i.e., $\mathcal{O}(V)$.

4.2 Theoretical Results

In this section, we supply an analysis of the proposed model, as well as a validation of the theoretical results. It is worth noting that only the main analytical differences in relation to theoretical analysis previously conducted on Section 3.3 (unsupervised version) are provided.

4.2.1 Mathematical Analysis

Since the dynamical system of the unsupervised and semisupervised version are virtually the same, differing only on the initial distributions of the particle locations, the transition probability function ought to remain the same. In view of this, we rewrite it for convenience matters as follows:

$$\begin{aligned}
 P(X(t+1) | X(t)) &= \mathbb{1}_{\{N(t+1)=N(t)+Q_N(p(t+1))\}} \\
 &\quad \times \mathbb{1}_{\{S(t+1)=Q_S(E(t+1))\}} \\
 &\quad \times \mathbb{1}_{\{E(t+1)=E(t)+\Delta Q_E(p(t+1), N(t+1))\}} \\
 &\quad \times \mathbb{P}_{\text{transition}}(N(t), p(t)) \\
 &= \mathbb{1}_{\{\text{Compliance}(t)\}} \mathbb{P}_{\text{transition}}(N(t), p(t)).
 \end{aligned} \tag{4.4}$$

Following the reasoning applied in the analytical analysis of the unsupervised dynamical system, we are required to set the feasible upper and lower limits of each stochastic variable. The limits for $p(t), E(t), S(t)$ derived in the unsupervised version are integrally valid for the semisupervised learning version.

Having in mind these considerations, we now derive these limits for the stochastic variable $N(t)$. The new initialization step indicated in (4.3) takes into account both labeled and unlabeled vertices, which invalidates Lemma 1 that only previews the existence of unlabeled vertices. Given this fact, we reformulate the aforementioned Lemma in the following.

* * *

Lemma 6. *The maximum reachable value of $N_i^{(k)}(t)$, $\forall(i, k) \in \mathcal{V} \times \mathcal{K}, t \in \mathbb{N}$, is:*

- If $i \in \mathcal{V}_u$:

$$N_{i_{\max}}^{(k)}(t) = \begin{cases} \left\lceil \frac{t+1}{2} \right\rceil + 1, & \text{if } t > 0 \text{ and } a_{ii} = 0 \\ t + 2, & \text{if } t > 0 \text{ and } a_{ii} > 0 \end{cases}; \quad (4.5)$$

- If $i \in \mathcal{V}_l$:

$$N_{i_{\max}}^{(k)}(t) = \infty. \quad (4.6)$$

Proof. With regard to unlabeled vertices, i.e., which belong to \mathcal{V}_u , the proof supplied in Lemma 1 can be invoked *ipsis litteris*. This is valid because the movement policy of each particle has remained the same in relation to the original unsupervised learning version.

With regard to labeled vertices, i.e., which belong to \mathcal{V}_l , this quantity can be inferred in a straightforward manner through the initial conditions of the system. According to (4.3), if i is a pre-labeled vertex and k is its representative particle, then $N_i^{(k)}(t) = \infty, \forall t \geq 0$. \square

* * *

Since the upper and lower limits of the stochastic variable $N(t)$ have changed, it is natural that the analysis of $\bar{N}(t)$ ought to suffer modifications, too. Similarly to the previous case, Lemma 3 presented in the original version of the algorithm may only be applied to unlabeled vertices. In view of this, we reformulate this Lemma as follows:

* * *

Lemma 7. *The following assertions hold $\forall (i, k) \in \mathcal{V} \times \mathcal{K}$:*

- If $i \in \mathcal{V}_u$:

(a) *The minimum value of $\bar{N}_i^{(k)}(t)$ is:*

$$\bar{N}_{i_{\min}}^{(k)}(t) = \frac{1}{1 + \sum_{u \in \mathcal{K} \setminus \{k\}} N_{i_{\max}}^{(u)}(t)}. \quad (4.7)$$

(b) *The maximum value of $\bar{N}_i^{(k)}(t)$ is:*

$$\bar{N}_{i_{\max}}^{(k)}(t) = \frac{N_{i_{\max}}^{(k)}(t)}{N_{i_{\max}}^{(k)}(t) + (K - 1)}. \quad (4.8)$$

- If $i \in \mathcal{V}_l$:

(a) The minimum value of $\bar{N}_i^{(k)}(t)$ is:

$$\bar{N}_{i_{\min}}^{(k)}(t) = 0. \quad (4.9)$$

(b) The maximum value of $\bar{N}_i^{(k)}(t)$ is:

$$\bar{N}_{i_{\max}}^{(k)}(t) = 1. \quad (4.10)$$

Proof. With regard to unlabeled vertices, the proof supplied in Lemma 3 can be invoked *ipsius litteris*.

With regard to labeled vertices, equation (4.9) can be reached as follows: consider that particle k is not a representative from the labeled vertex i . However, by the initial conditions shown in (4.3), since vertex i is labeled by hypothesis, $\exists k' \in \mathcal{K} : N_i^{(k')}(t) = \infty, \forall t \geq 0$. Now, as k does not represent i , via (4.3) again, we know that $N_i^{(k)}(t)$ may only take on finite values $\forall t \geq 0$. Finally, applying (3.7) using this setup yields (4.9). Equation (4.10) can be achieved as follows: consider that particle k now represents the labeled vertex i . In this case, $N_i^{(k)}(t) = \infty$. Considering that a labeled vertex may only be represented by one kind of particle, then all the remaining entries of $N_i^{(k')}(t), k' \in \mathcal{K}, k' \neq k$ are finite. Using (3.7) under these circumstances, we arrive at (4.10).

□

* * *

The final step before calculating the marginal distribution of the vertices' domination levels, i.e., $P(\bar{N}(t))$, is to find all the possible irreducible fractions that an arbitrary entry of $\bar{N}(t)$ can assume. Lemma 4 fails to provide us with enough information about the labeled vertices, since it only delimits the irreducible fractions for unlabeled vertices. Next, a reformulation of such Lemma is provided.

* * *

Lemma 8. Denote num/den as an arbitrary irreducible fraction. Consider that the set \mathcal{I}_t retains all the reachable values of $\bar{N}_i^{(k)}(t)$, $\forall (i, k) \in \mathcal{V} \times \mathcal{K}$, for a fixed t . Then, the elements of \mathcal{I}_t are composed of all elements satisfying the following constraints:

(i) With regard to unlabeled vertices:

(a) The minimum element is given by the expression in (3.56).

(b) The maximum element is given by the expression in (3.57).

(c) All the irreducible fractions within the interval delimited by (a) and (b) such that:

$$\text{I. } \text{num}, \text{den} \in \mathbb{N}^*.$$

$$\text{II. } \text{num} \leq N_{i_{\max}}^{(k)}(t)$$

$$\text{III. } \text{den} \leq \sum_{u \in \mathcal{K}} N_{i_{\max}}^{(u)}(t)$$

(ii) With regard to labeled vertices:

(a) 0, if particle k does not represent vertex i .

(b) 1, if particle k represents vertex i .

Proof. Regarding item (i): Straightforward from Lemma 4.

Regarding item (ii): (a) As vertex i is labeled, $\exists u \in \mathcal{K} : N_i^{(u)}(t) = \infty$. In view of (3.7) and (4.3), we obtain $\bar{N}_i^{(k)}(t) = 0$; (b) Similarly, using (3.7) and (4.3), we get $\bar{N}_i^{(k)}(t) = 1$. \square

* * *

Finally, the expression for calculating the domination matrix distribution remains the same as the one derived in the unsupervised version of the algorithm. In the following, we reiterate it for convenience:

$$P(\bar{N}(t) = U : U \in \mathcal{M}_t) = \sum_{u=1}^t P(f(uN(t)) = U). \quad (4.11)$$

As $t \rightarrow \infty$, $P(\bar{N}(t))$ provides enough information for classifying the unlabeled vertices. In this case, they are labeled according to the team of particles that is imposing the highest domination level. Since the domination level is a normalized stochastic variable, the output of this model is fuzzy.

4.2.2 A Numerical Example

In this section, we provide how the theoretical results supplied in the previous section can be used. We limit the demonstration for a single iteration, namely for transition from $t = 0$ to $t = 1$. In order to do so, a simple example composed of a trivial 3-vertex regular network, identical to the one in Fig. 3.12a, is analyzed. In this network, vertex 1 has been labeled as pertaining to class 1 and vertex 2, to class 2, i.e., $\mathcal{V} = \{1, 2, 3\}$, $\mathcal{V}_L = \{1, 2\}$, and $\mathcal{L} = \{1, 2\}$. Clearly, vertex 3 possesses overlapping characteristics in relation to classes 1 and 2. We will theoretically show this behavior through this illustrative example. Consider the arbitrary initial settings: we insert $K = 2$ particles into the network, i.e., $\mathcal{K} = \{1, 2\}$. Let the particle 1 represent vertex 1 (i.e., it will propagate the label of vertex 1) and particle 2, vertex 2. Suppose also that we have a certainty about the locations of the particles at $t = 0$, which satisfy the following distribution:

$$P \left(N(0) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 1 & 1 \end{bmatrix}, p(0) = [1 \ 2], E(0), S(0) \right) = 1, \quad (4.12)$$

i.e., there is an 100% (certainty) that the particles 1 and 2 are generated at vertices 1 and 2, respectively. Observe that $N(0)$, $E(0)$, and $S(0)$ are chosen such as to satisfy (4.3), (3.24), and (3.25), respectively; otherwise the probability should be 0, in view of (4.4).

From Fig. 3.12a we can deduce the adjacency matrix A of the graph and, therefore, determine the transition matrix associated to the random movement term for one particle (which is the same to the other particle). Then, applying (3.2), we get:

$$\mathbb{P}_{\text{rand}} = \begin{bmatrix} 0 & 0.50 & 0.50 \\ 0.50 & 0 & 0.50 \\ 0.50 & 0.50 & 0 \end{bmatrix}. \quad (4.13)$$

Given $N(0)$, we can readily establish $\bar{N}(0)$ with the aid of (3.7):

$$\bar{N}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.50 & 0.50 \end{bmatrix}. \quad (4.14)$$

Using (3.8) we are able to calculate the matrices associated to the preferential move-

ment policy for each particle in the network:

$$\mathbb{P}_{\text{pref}}^{(1)}(0) = \begin{bmatrix} 0 & 0 & 1 \\ 0.67 & 0 & 0.33 \\ 1 & 0 & 0 \end{bmatrix}, \quad (4.15)$$

$$\mathbb{P}_{\text{pref}}^{(2)}(0) = \begin{bmatrix} 0 & 0.67 & 0.33 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (4.16)$$

In order to ease the calculations, let us take $\lambda = 1$, so that (4.1) reduces to $\mathbb{P}_{\text{transition}}(0) = \mathbb{P}_{\text{pref}}^{(1)}(0) \otimes \mathbb{P}_{\text{pref}}^{(2)}(0)$ at time 0, which will be a matrix with dimensions 9×9 . Instead of building this matrix, we will make use of Remark 3 to build the next particles localization vector $p(1)$ with the collection of two matrices 3×3 , as given in (4.15) and (4.16). Note that, in the special case when $\lambda = 1$, the preferential movement matrix is the transition matrix itself, provided that all the particles are active, which indeed are at time 0, according to (3.25). For the first particle, one can see from (4.15) that, starting from vertex 1 (row 1), there can only be one next possible localization for particle 1, namely vertex 3. For the second particle, starting from the vertex 2 (row 2), one can state that the next localization of particle 2 can only be vertex 3, too. With that in mind, we have that:

$$P \left(N(1) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 2 & 2 \end{bmatrix}, p(1) = [3 \ 3], E(1), S(1) \mid X(0) \right) = 1, \quad (4.17)$$

where $X(0)$ is given by (4.12). Furthermore, as we have fixed $\lambda = 1$, it is expected that the transition will be heavily dependent on the domination levels of the neighborhood vertices. Therefore, given that the labeled vertices constitute strong repulsive forces that act against rival particles, the preferential or biased behavior of these particles will never adventure in these type of vertices. This provides a natural explanation for the reason that the state $p(1) = [3 \ 3]$ is the only possible next particles localization vector.

Before doing the calculation of the marginal distribution $P(N(1))$, we are required to find an upper limit for an arbitrary entry of a specific unlabeled vertex of the matrix $N(1)$. This is readily evaluated from (4.5), which results in $N_{i_{\max}}^{(j)}(1) = 2, \forall i \in \mathcal{V}$, implying that we are only needed to take all numerical combinations of the matrix $N(0)$ such that each entry may only take the values $\{1, 2\}$, since larger values would yield probability 0 according to Lemma 6. Moreover, we need to iterate through every

feasible value of every entry of $E(0)$ and $E(1)$. In order to do so, we fix $\Delta = 0.25$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. With that, we are able to make use of Lemma 2, which yields $E(t) \in \{0, 0.25, 0.5, 0.75, 1\}$. The limits of the remaining system variables $S(0)$ and $S(1)$ are straightforward. In the present conditions, we have enough information to calculate the marginal distribution $P(N(1))$, according to (3.53):

$$P\left(N(1) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 2 & 2 \end{bmatrix}\right) = 1 \times 1 = 1. \quad (4.18)$$

As the last goal, our task is to determine the distribution $P(\bar{N}(1))$. According to the specified steps in the previous section, we need to find all irreducible fractions that lie within the interval $[0, 1]$ with the constraints derived in the previous section. This means that we only have to consider entries of matrix $\bar{N}(t)$ that contain elements of \mathcal{I}_t ; the remainder $\bar{N}(t)$ are infeasible and, thus, occur with probability 0. In view of the constraints previously enumerated, $\mathcal{I}_t = \{0, 1/4, 1/3, 1/2, 2/3, 3/4, 1\}$. It is worth commenting that the labeled vertices (vertices 1 and 2) can only assume the values $\{0, 1\} \subset \mathcal{I}_t$, as we have previously stated. Observing that we have the complete distribution of $N(1)$, it is an easy task to apply (3.60), as follows:

$$P\left(\bar{N}(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix}\right) = 1. \quad (4.19)$$

It is noteworthy to reinforce that the mapping between the probabilities of $N(t)$ and $\bar{N}(t)$ is not bijective: in this special simple case that we are studying, we did not have distinct $N(t)$ that could generate $\bar{N}(t)$, but as t increases, this is likely to happen quite frequently. This process is repeated for a sufficiently large t or until the system converges to a quasi-stationary state $\bar{N}(t)$. A detailed look at the system's behavior that we have derived suggests that (4.19) holds for every $t \geq 1$, and particles 1 and 2 will visit vertex 3 with period 2. Hence, this shows the overlapping nature of vertex 3, as it can be naturally stated only by the topological structure of the graph. Ideally, for networks with presence of distinct classes, $\bar{N}(t)$ will change. Specifically, the $\bar{N}(t)$ with the highest probability will be the one responsible for unveiling the most probable classification of each vertex in the given network.

4.2.3 Validation of the Theoretical Results

In this section, we will show that the theoretical results presented in the previous section approximate the empirical behavior of the stochastic competitive model for a large number of independent runs of the algorithm. In order to do so, we will utilize the network shown in Fig. 3.3, i.e., $\mathcal{V} = \{v_1, \dots, v_{20}\}$. We label two vertices, namely v_6 with the red or dark gray label and v_{18} with the blue or light gray label. We insert two particles $K = \{1, 2\}$, where particle 1 is initially located at vertex v_6 and will be the carrier and the propagator of the red or dark gray label, whereas particle 2 is initially placed at vertex v_{18} and will be responsible for propagating the blue or light gray label.

For the empirical evaluation, since the proposed competitive model is stochastic, in order to estimate the empirical domination level matrix $\bar{N}(t)$, we will run the algorithm 10 000 times in an independent manner. For each run, we iterate the system until $t = 1\,000$ and store the domination levels imposed by every particle in the network on every vertex, i.e., we maintain the matrix $\bar{N}(1000)$ generated by the stochastic dynamical system for all the 10 000 runs. Once we have calculated all the matrices $\bar{N}(1\,000)$ associated to all the 10 000 independent runs, we build up $V \times K$ histograms, in which each histogram represents an entry of $\bar{N}(1000)$, and populate it according to the value $\bar{N}_i^{(k)}(1\,000), \forall (i, k) \in \mathcal{V} \times \mathcal{K}$, reached at every run. For example, we put all the domination levels imposed by the red or dark gray particle on the vertex v_1 , i.e., $\bar{N}_1^{(\text{red})}(1\,000)$, in a histogram for each run. Since the domination levels are continuous values in the interval $[0 - 1]$, we discretize the interval $[0 - 1]$ using bins with 0.01 width each, i.e., 100 bins. At the end, we normalize each histogram by dividing by the number of independent runs (10 000), so we have an estimated probability of the domination level matrix on that vertex for a specific particle. We do this process for every particle and vertex in the network.

Regarding the theoretical evaluation of the network in Fig. 3.3, we use the same parameters as the empirical evaluation, so we can compare the validity of our results previously shown. In order to get the theoretical distribution of the domination level matrix $\bar{N}(1\,000)$, we directly apply (3.60). With that, we are able to evaluate the probability of occurring a specific domination level matrix $\bar{N}(1\,000)$. Since we can not plot the probability distribution of this matrix, on account of being in the space $V \times K + 1$, we marginalize the distribution on three specific vertices, so as to be able to graphically demonstrate the resulting distribution. We perform this process on vertices v_4 (a member of the red or dark gray class), v_{11} (a vertex in the border of both classes), and v_{16} (a member of the blue or light gray class). Figure 4.2 shows the estimated probability distribution of the domination level imposed by the red or dark gray particle on these three vertices (blue or light gray curve) and also depicts the obtained theoretical distribution (red or dark gray curve). As one can see by Fig. 4.2a, v_4 is almost com-

pletely dominated by the red or dark gray particle, since the domination level imposed by this particle approximates 1 (and, consequently, the domination level imposed by the blue or light gray particle on the same vertex decays to 0). Figure 4.2b confirms the overlapping nature of v_{11} , since the domination level of both particles red or dark gray and blue or light gray are almost the same, and Fig. 4.2c indicates that the red or dark gray particle has little domination on v_{16} (therefore, the blue or light gray particle has almost complete domination on this vertex). These curves must be interpreted in the following manner: take Fig. 4.2a for instance, there is an approximately 34% chance that the domination level imposed by the red or dark gray particle on v_4 to be 0.88 if we start the system at time $t = 0$ and stop evolving it at $t = 1000$. Other values are possible, but with rarer chances. As we can visually verify, the theoretical results have approximately matched the results obtained by the empirical computer simulations, confirming our theoretical analysis.

4.3 Computer Simulations

In this section, the performance of the algorithm will be tested against several types of distributions of data items. Additionally, a study on the parameters of the model is described. This investigation is important because it enables one to better understand in which situations the proposed algorithm is a good candidate and when it is not.

4.3.1 Parameter Sensitivity Analysis

We start out this section by analyzing the behavior of the parameter λ , which is responsible for counterweighting the proportion of preferential and random walks performed by all particles in the network. In this section, we will study its impact on random clustered networks, whose construction has already been described in the previous section. We set $\Delta = 0.07$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. Figure 4.3 shows how the accuracy of the model behaves as we vary λ from 0 (pure random walks) to 1 (pure preferential walks). As one can verify from the figure, this parameter is sensible to the outcome of the technique. Usually, the optimal accuracy is reached when a mixture of random and preferential walks occurs. Specifically, for $0.2 \leq \lambda \leq 0.8$, the model gives good accuracy results.

Another important parameter that needs to be addressed is Δ . It is responsible for updating the energy of the particles, according to which type of vertex a specific particle is visiting. We will apply the same networks as used in the previous analysis, and we will fix $\lambda = 0.6$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$ for this simulation. Figure 4.4 portrays the accuracy reached by the algorithm for varying values of Δ . We can see that for intermediate values of Δ , namely $0.05 < \Delta < 0.4$, the model is not sensitive. However,

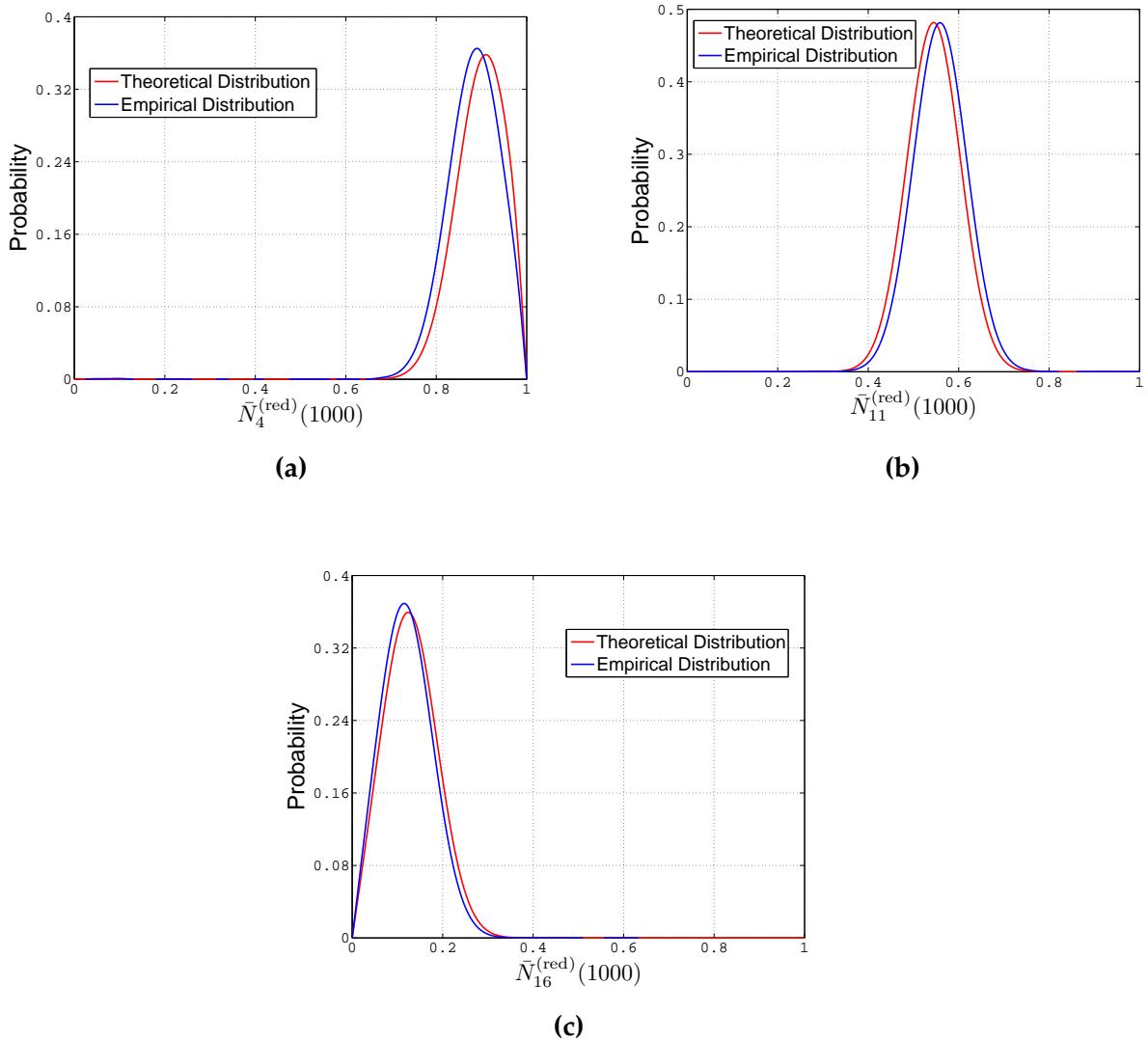


Figure 4.2: Comparison of the theoretical and empirical distributions of three distinct vertices, namely, v_4 , v_{11} , and v_{16} . The curves have suffered minor filtering (smoothing), so as to become more readable. (a) One can see that the most probable domination level that the red or dark gray particle will impose on v_4 is approximately 0.88 with 34% probability, (b) on v_{11} is 0.53 with 47% probability, and (c) on v_{16} is 0.14 with 33% probability.

as the value gets higher and higher, the performance of the technique is reduced. This happens because the particles get exhausted as soon as they visit a vertex from the rival particle. In this way, it becomes hard for the particles to change the ownership of previously conquered vertices. We can understand this process as an artificial “hard labeling.” On the other hand, for Δ too low, the particles are free to travel around the network with no energy penalties, i.e., they will rarely get exhausted. Therefore, all vertices in the network will be in constant competition and no class borders will be established.

It is worth stressing that ω_{\min} and ω_{\max} need not to be analyzed, since they are

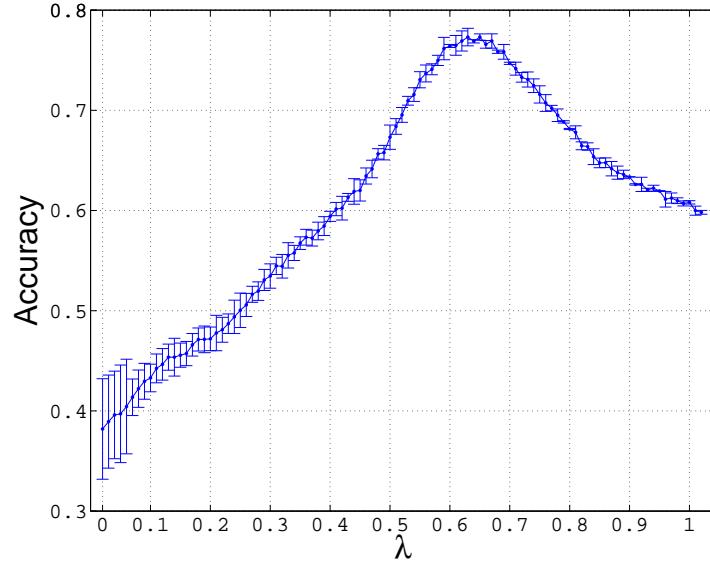


Figure 4.3: Classification accuracy vs. λ . In these simulations, $N = 1000$, there are 4 equal sized classes, $\langle k \rangle = 16$, and $z_{\text{out}}/\langle k \rangle = 0.4$. $\Delta = 0.07$. Each point in the trace is averaged by 100 realizations. The error bars represent standard deviations.

simply defining an interval. Furthermore, Δ is not a sensitive parameter, because the interval of Δ to get good results is very large. In view of all parameter sensitivity examination, in the following simulations, we will always fix $\Delta = 0.07$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$.

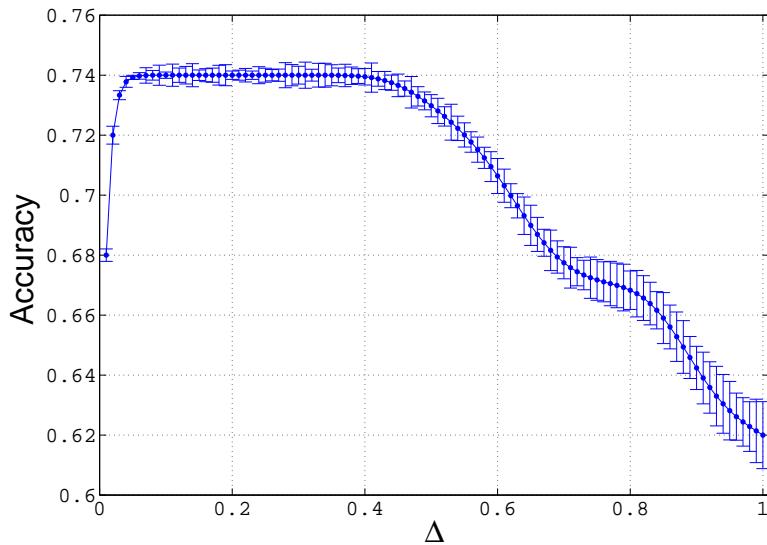


Figure 4.4: Classification accuracy vs. Δ . In these simulations, $N = 1000$, there are 4 equal sized classes, $\langle k \rangle = 16$, and $z_{\text{out}}/\langle k \rangle = 0.4$. $\lambda = 0.6$. Each point in the trace is averaged by 100 realizations. The error bars represent standard deviations.

4.3.2 Simulations on Synthetic Data Sets

In order to facilitate the understanding of how the proposed technique works, we will first look at two simple semisupervised classification problems. Afterwards, we will look at more elaborated problems, where the classes have irregular forms.

A Simple Balanced Two-class Problem

Here, we design a synthetic data set with two simple classes, each of which with 50 vertices. We have inserted into the network 2 particles, each one representing a class. With this simple data set, we are able to closely observe the behavior of the proposed algorithm. Figure 4.5a shows the initial configuration of the network, where the colored (shades of light gray) dots symbolize labeled samples. The black (dark gray) dots denote unlabeled data. The ownership of any vertex is given by the particle which has the highest domination level over that vertex. For this simulation, we use $\lambda = 0.6$. According to (4.3), the vertices that are initially labeled have their ownership fixed to their corresponding representative particles. As the dynamical system evolves, the particles will visit the vertices of the network in agreement with the probability distribution given by the matrix $\mathbb{P}_{\text{transition}}(t)$. Figure 4.5b shows the ownership of each vertex after 100 iterations, Fig. 4.5c depicts the ownership state after 200 iterations, and Fig. 4.5d reveals the quasi-stationary state of the algorithm $\bar{N}(t)$, which is reached after 300 iterations. We now take a closer look at the evolutional behavior of the average domination level of the vertices that belong to the same class. Figure 4.6a provides the average domination level imposed by the particle representing the initially blue (labeled vertex at the class in the left in Fig. 4.5a) labeled vertex on the vertices 1 to 50 (blue class) and 51 to 100 (red class), whereas Fig. 4.6b displays the same information for the particle representing the initially red (labeled vertex at the class in the right in Fig. 4.5a) labeled vertex. Clearly, as time progresses, one can see that the classes are unmistakeably separated by the competitive system.

A Simple Unbalanced Three-class Problem

Here, we investigate the performance of the algorithm when applied to a network consisted of $V = 15$ vertices split into 3 unbalanced communities, as depicted in Fig. 4.7. $K = 3$ particles are inserted into the network at the initial positions $p(0) = [2 \ 8 \ 15]$, meaning the first particle (representing the red or “circle” class) starts at vertex 2, the second particle (representing the blue or “square” class) starts at vertex 8, and the third particle (representing the green or “triangle” class) starts at vertex 15. All the remaining vertices in the network are initially unlabeled (in the figure, they are colored for the sake of easily identifying the classes). The competitive system is iterated until $t = 1\,000$ and the predicted label for each of the unlabeled vertices is given by the particle’s la-

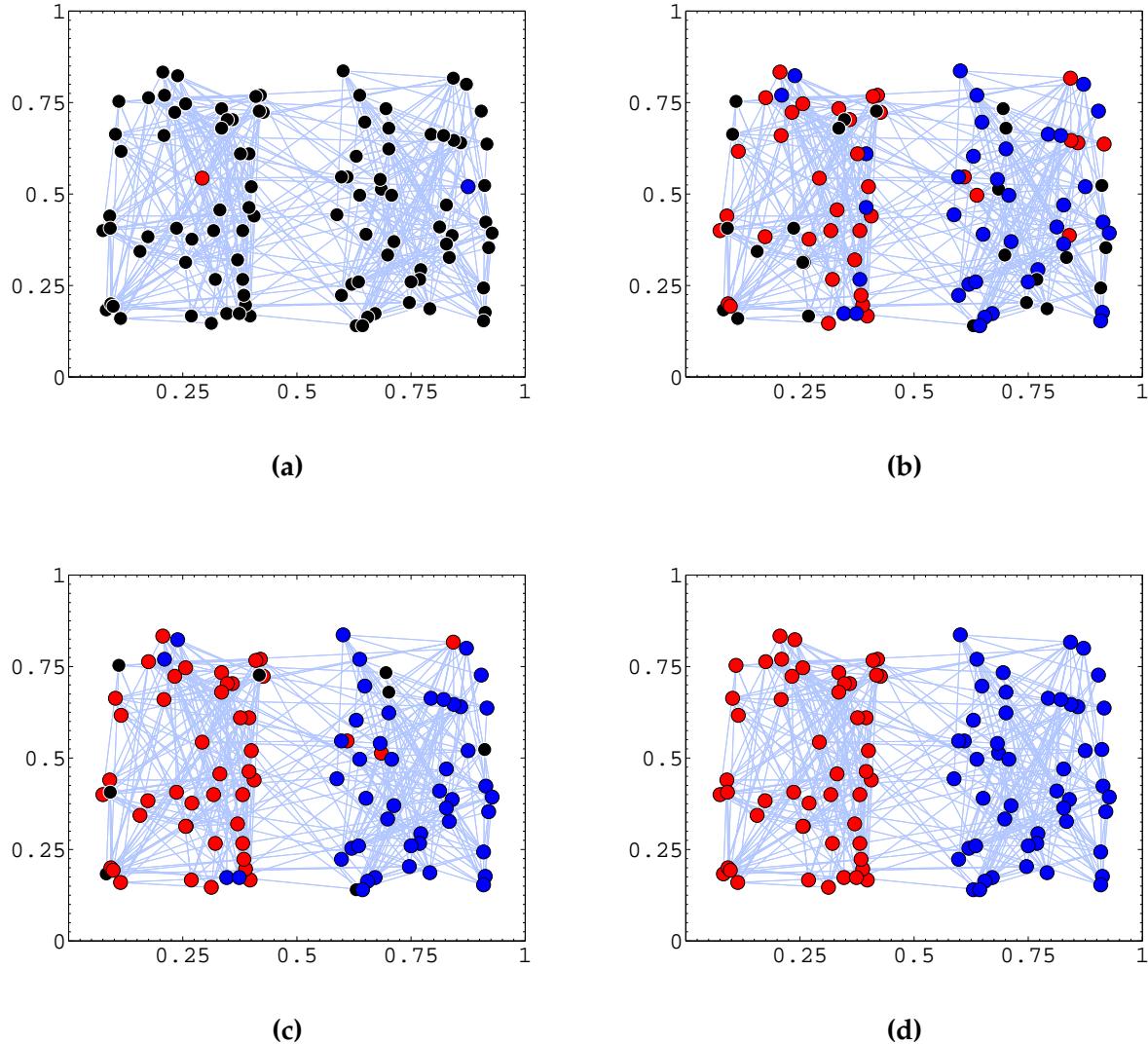


Figure 4.5: Illustration of an artificial classification process through competitive particle walking. The total number of vertices is $V = 100$. Vertices 1 to 50 are present in the left and vertices 51 to 100, in the right. (a) A snapshot of the initial configuration: there are two previously labeled vertices (red and blue vertices) and $K = 2$ particles, each one representing a labeled vertex. The black dots represent unlabeled data. The particles have been spawned at their representative labeled vertex. (b) A snapshot at iteration 100. (c) A snapshot at iteration 200. (d) A snapshot at iteration 300.

bel that is imposing the highest domination level. Figures 4.8a, 4.8b, and 4.8c show the evolutional behavior of the domination levels imposed by the three particles on the red or “circle” class, the blue or “square” class, and the green or “triangle” class, respectively. Specifically, from Fig. 4.8a, we can verify that red or “circle” particle dominates vertices 1 to 4 (red or “circle” class), due to the fact that the average domination level on these vertices approaches 1, whereas the average domination levels of the other two rival particles decay to 0. Considering Figs. 4.8b and 4.8c, we can use the same logic to confirm that the blue or “square” particle completely dominates the vertices 5 to 10

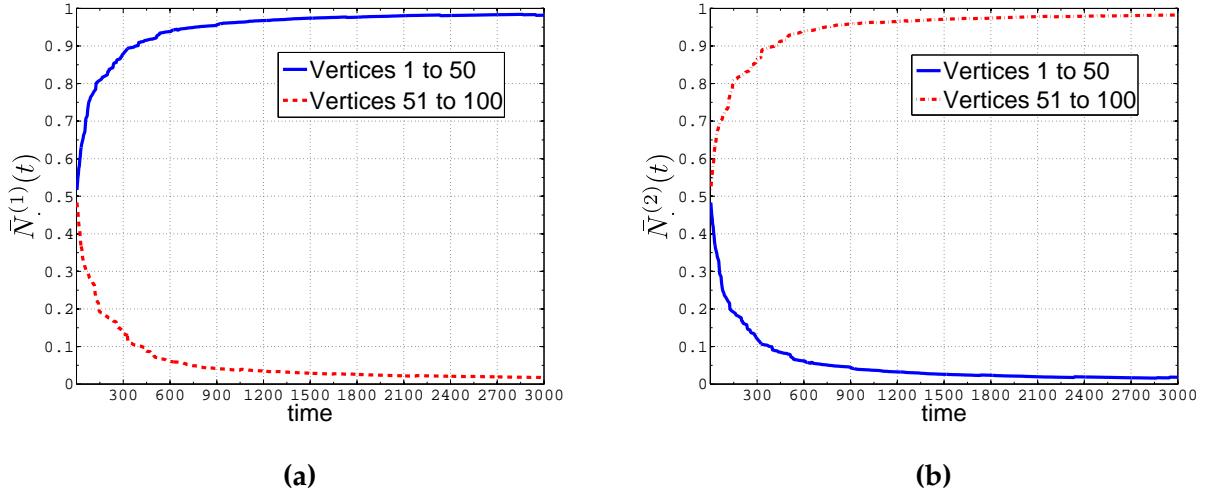


Figure 4.6: Evolutional behavior of the average class domination level imposed by the particles in the network. (a) Average class domination level imposed by particle 1. (b) Average class domination level imposed by particle 2.

(blue or “square” class) and the green or “triangle” particle dominates vertices 11 to 15 (green or “triangle” class).

Data Sets with Irregular Forms

Now we proceed to test the proposed technique on data classes with arbitrary forms of distributions. In order to do so, we make use of some toy data sets with different data distributions, which are automatically generated by PRTools (Duijn, 2000). For all simulations taken on artificial data, we have arbitrarily chosen the k -nearest neighbor graph formation technique with $k = 5$, i.e., each data item is represented by a vertex and each vertex is connected to its 5 nearest neighbors determined by the Euclidean distance among the data items. For each 50 vertices that are unlabeled, we randomly choose one amongst them and label it. For each labeled vertex, we generate one representative particle. Note that in this case, there could be more than 1 particle representing the same team. The first data set, as shown in Fig. 4.9a, consists of 600 samples equally divided into two banana-shaped classes. The result is exhibited at Fig. 4.9b. The second data set, which can be seen at Fig. 4.9c, is composed of 600 samples equally divided into two Highleyman classes. The corresponding outcome is provided at Fig. 4.9d. The third data set, as depicted by Fig. 4.9e, comprises 550 samples separated into two Lithuanian classes of sizes 250 and 300 vertices. The corresponding result is supplied in Fig. 4.9f. The forth data set, as can be visualized in Fig. 4.9g, is made up of 800 samples equally divided into four 2D Gaussian-distributed classes. Its corresponding outcome is portrayed in Fig. 4.9h. All results seem to be visually satisfactory, according to the provided inputs, reinforcing the robustness of the technique to detect arbitrary

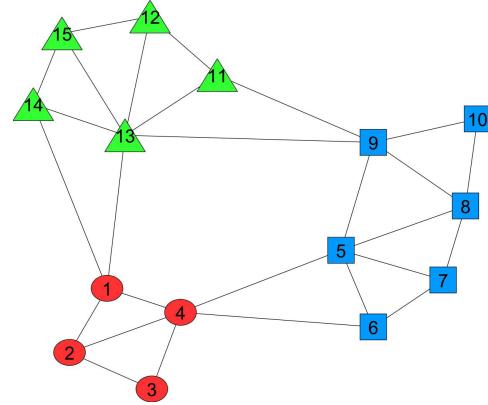


Figure 4.7: A simple networked data set. The red or “circle” class is composed by vertices 1 to 4, the blue or “square” class comprises the vertices 5 to 10, and the green or “triangle” class encompasses the vertices 11 to 15. Initially, only the vertices 2 (red or “circle” agent), 8 (blue or “square” agent), and 15 (green or “triangle” agent) are labeled.

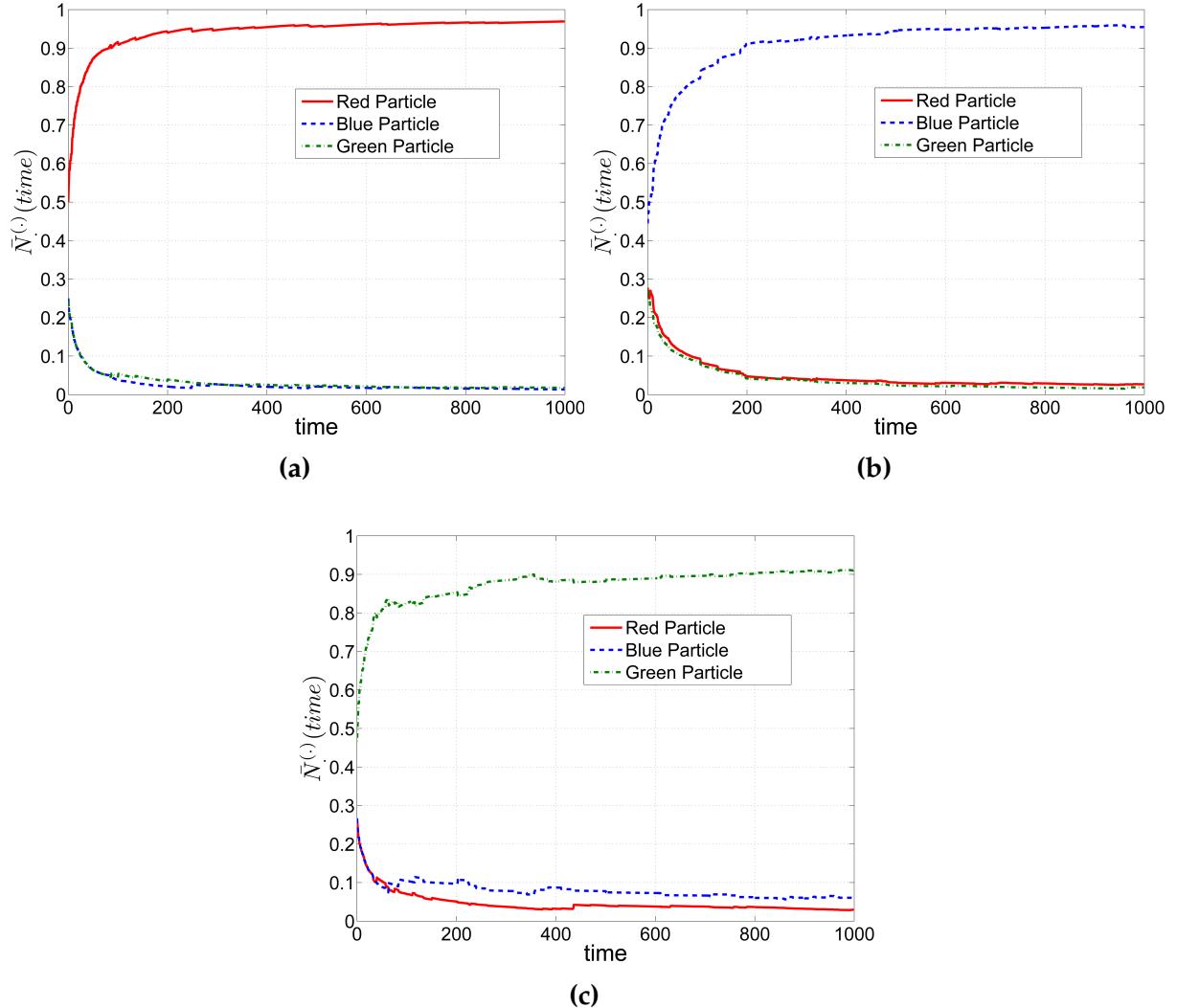


Figure 4.8: Evolutional behavior of the average domination level imposed by the 3 particles in the network on: (a) the red or “circle” class (vertices 1 to 4), (b) the blue or “square” class (vertices 5 to 10), and (c) the green or “triangle” class (vertices 11 to 15).

form of class distributions.

4.3.3 Computer Simulations on Benchmarked Data Sets

In this section, we provide external comparable results by using well-known data sets, which have become benchmarks in the machine learning community. We start out by the broad accepted data sets of the UCI Machine Learning Repository. Afterwards, we delve into the recent network-based benchmark proposed by Lancichinatti *et. al.* ([Chapelle *et al.*, 2006](#)).

UCI Machine Learning Repository

In this section, computer simulations on real-world data sets gathered from the UCI Machine Learning Repository are presented. A brief meta-information of the 14 real-world data sets that are going to be studied here is supplied in Table 4.1. For a detailed description, refer to ([Frank and Asuncion, 2010](#)).

Table 4.1: Brief meta-information of the UCI data sets used in the semisupervised learning process.

Data Set	# Instances	# Dimensions	# Classes
Heart	303	75	2
Heart-Statlog	270	13	2
Ionosphere	351	34	2
Vehicle	946	18	4
House-Votes	435	16	2
Wdbc	569	32	2
Clean1	476	168	2
Isolet	7 797	617	26
Breast Cancer	569	32	2
Australian	690	14	2
Diabetes	768	8	2
German	1 000	20	2
Optdigits	5 620	64	10
Sat	6 435	36	7

With respect to the experimental setup, for each data set, 10 examples are randomly selected to compose the labeled set and the remainder of the examples are left unlabeled. The labeled sets are designed in such a way that each class retains, at least, a labeled example. The experiments are repeated 30 times for each labeled set and the average test error and standard deviations are recorded. The error estimative is quantified by using a hold-out technique in the evaluation process. Also, for the purpose of comparing the performance of different algorithms, we use two competitive semisupervised data classification techniques: Transductive SVM (TSVM) and Low Density

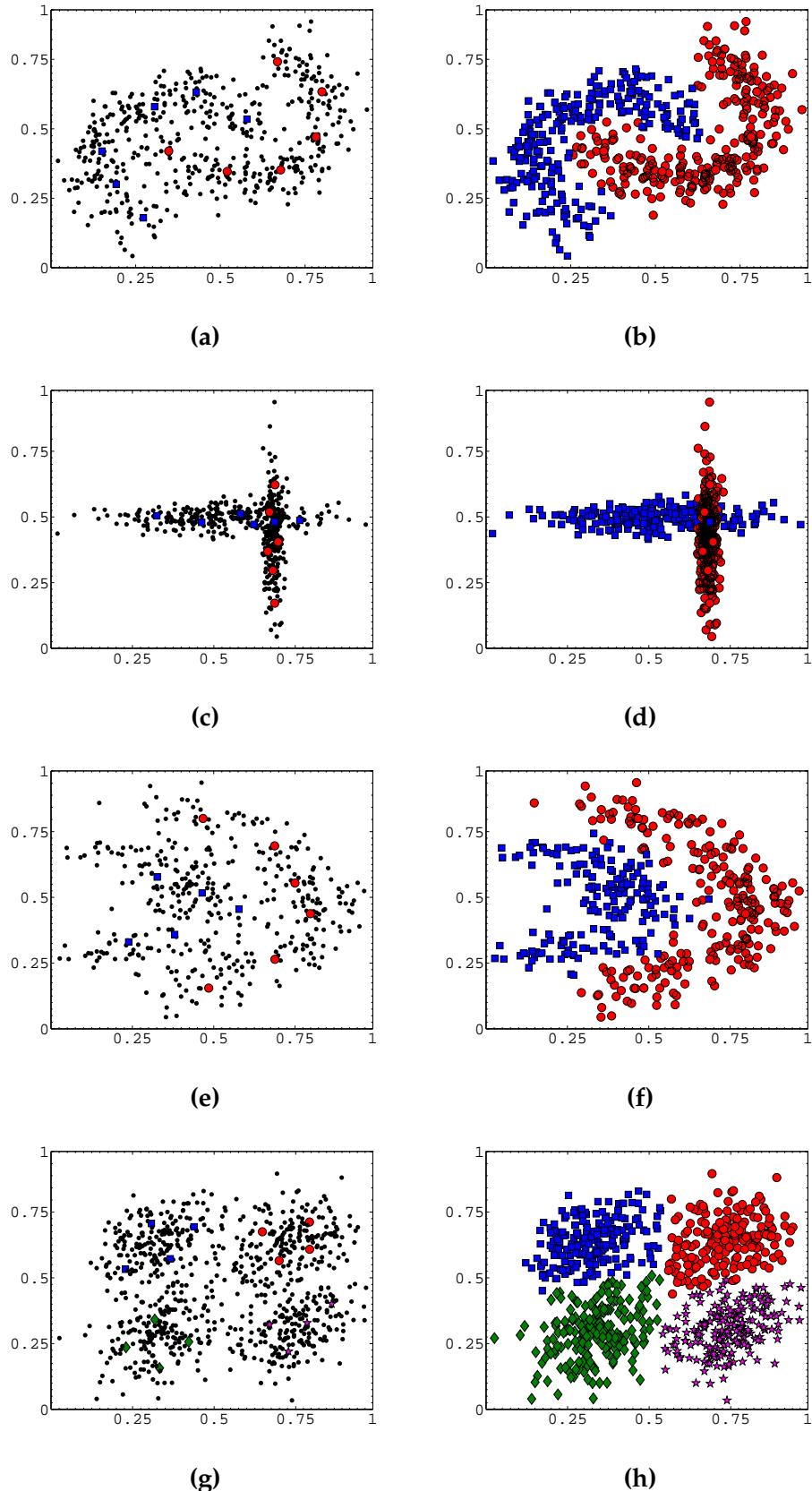


Figure 4.9: Data Classification of toy data sets with: (a) 600 samples equally divided into two banana-shaped classes; (b) 600 samples equally divided into two Highleyman classes; (c) 550 samples divided into two Lithuanian classes of sizes 250 and 300 vertices; (d) 800 samples equally divided into four Gaussian-distributed classes. Data with the same color represent the same class. The black dots depict unlabeled vertices. The colored dots in the left column represent previously labeled data.

Separation (LDS). For a description and the default parameters used, one can refer to ([Chapelle et al., 2006](#)).

Regarding the proposed technique, a graph formation technique must be employed. For this end, the k -nearest neighbor technique with $k = 3$ is utilized. The following parameters are fixed for all the data sets in this section: $\lambda = 0.6$, $\Delta = 0.07$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. Usually, the model provides good results when λ is within the interval $[0.2, 0.8]$, meaning that a combination of random-preferential walk is responsible for producing better results.

The simulation results are reported in Table 4.2. In this table, the average rank of each algorithm is provided, whose calculation method is described in the following:

- i. For each data set, the algorithms are ranked according to their average performance, i.e., the best algorithm (the smallest test error) is ranked as 1st, the second best one is ranked as 2nd, and so on;
- ii. For each algorithm, the average rank is given by the average value of its ranks scored on all the data sets.

With the purpose of examining these simulation results in a statistical manner, the procedure outlined in ([Demšar, 2006](#)) is adopted. Basically, the methodology comprises two steps:

- *First Step:* A Friedman test is applied to check whether the algorithms under comparison are significantly distinct from each other. Such a test is used to detect differences in treatments across multiple test attempts. Instead of doing pairwise comparisons, this test aims at discovering whether all the techniques present any statistical difference. The procedure involves ranking each row together, then considering the values of ranks by columns. It basically compares a statistic descriptor in relation to a critical value taken from an F-distribution. Note that this descriptor is global: if its value surpasses this critical value, then it is said that the set of algorithms presents statistical difference in their performances. The null hypothesis here is that all algorithms present similar performances. However, with such test, we do not know which of these techniques are superior to the others. Thus, the second step, which utilizes the Bonferroni-Dunn test, is introduced to fulfill this gap;
- *Second Step:* In the case that the performances of the involved algorithms are different, then we apply post hoc tests by using the Bonferroni-Dunn test, with the objective to verify whether the performance of the control algorithm is really superior to the others. In this case, the technique proposed in this paper is set as the control algorithm. Basically, the Bonferroni-Dunn Test quantifies whether

the performance between an arbitrary algorithm and the reference is significantly different. This is done by verifying whether the corresponding average ranks of these algorithms differ by at least a critical difference (CD). If they do differ that much, then it is said that the better ranked algorithm is statistically superior to the worse ranked one. Otherwise, they do not present a significant difference for the problem at hands.

The reason behind choosing this particular suite of statistical tests is that, as ([Demšar, 2006](#)) draws attention to, a set of non-parametric statistical tests is more convenient when comparing a set of classifiers over multiple data sets, which is the case of the Friedman Test. In addition, it has been shown that the non-parametric tests are more likely to reject the null-hypothesis, which hints at the presence of outliers or violations of assumptions of the parametric tests. Moreover, the Friedman test is appropriate in these circumstances because it assumes some, but limited commensurability. It is safer than parametric tests, since it does not assume normal distributions or homogeneity of variance. As such, it can be applied to compare classification accuracies, error ratios or any other measure for evaluation of classifiers, including even model sizes and computation times. However, the Friedman test can only provide global information, i.e., it does not point which of the algorithms are statistically superior to the others. With this in mind, we have used the Bonferroni-Dunn test to statistically perform these pairwise comparisons, given that the set of algorithms as a whole present a statistical significance. It has been shown that this filtering before using the Bonferroni-Dunn test leads to more robust statistical conclusions ([Demšar, 2006](#)).

In our experiments in the UCI data sets, we fix a significance level of 0.05. For our experiments, according to ([Demšar, 2006](#)), we have that $N = 14$ and $k = 3$, resulting in a critical value given by $F(2, 26) \approx 3.37$, where the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. In our case, we get a value $F_F \approx 5.32$ that is higher than the critical value, so the null-hypothesis is rejected at a 5% significance level.

As the null hypothesis is rejected, one can advance to post hoc tests which aim at verifying the performance of the proposed algorithm in relation to others by employing the Bonferroni-Dunn Test. In this context, the control algorithm is fixed as the proposed technique. Thus, if we perform the evaluation of the CD for our problem, we encounter $CD \approx 0.8$. The average rank of the proposed method is 1.6. By virtue of that, if any rank does lie in the interval 1.6 ± 0.8 , the control algorithm and the compared algorithms are statistically equivalent. We conclude that our algorithm is superior to Transductive SVM for the simulations performed on these data sets. However, the comparison of the LDS to the control algorithm does not surpass the CD, meaning that the differences among them are statistically insignificant. Nonetheless, the proposed

technique has obtained the best performance (the best average rank) in relation to the other techniques.

Table 4.2: Test errors (%) with 10 labeled training points and the corresponding average rank and standard deviation of each technique.

Data Set	TSVM	LDS	Proposed Technique
Heart	27.4 ± 10.4	22.9 ± 9.6	21.3 ± 9.9
Heart-Statlog	26.1 ± 5.9	21.7 ± 6.1	20.5 ± 5.4
Ionosphere	23.9 ± 8.2	24.1 ± 10.9	24.7 ± 9.0
Vehicle	36.8 ± 7.8	33.7 ± 8.5	31.8 ± 8.8
House-Votes	16.0 ± 5.3	11.6 ± 4.0	11.4 ± 3.7
Wdbc	11.1 ± 3.7	15.0 ± 8.7	11.9 ± 5.1
Clean1	46.7 ± 4.8	43.2 ± 3.7	40.2 ± 2.9
Isolet	13.3 ± 9.5	8.0 ± 11.4	12.7 ± 8.8
Breastw	11.1 ± 8.8	9.6 ± 7.6	10.5 ± 9.4
Australian	31.4 ± 11.4	34.0 ± 14.5	31.6 ± 12.2
Diabetes	34.2 ± 4.6	33.8 ± 4.8	32.1 ± 4.6
German	36.5 ± 5.1	35.3 ± 4.2	35.9 ± 4.3
Optdigits	8.6 ± 7.6	3.6 ± 11.1	5.4 ± 8.9
Sat	13.5 ± 10.8	5.8 ± 14.2	9.3 ± 10.1
Average Rank	2.6	1.9	1.6

Network-based Benchmark of Chapelle et. al.

In order to measure the performance of the proposed method, we have carried out experiments to 7 standard semisupervised data sets in the transductive setting, i.e., the test set coincides with the set of unlabeled points. Each data set was constructed in an attempt to create situations that correspond to certain assumptions that semisupervised learning algorithm usually are based on, which are: smoothness assumption, cluster assumption, and the manifold assumption. The other 4 data sets are derived from real data. Since it is important to know the characteristics of the data sets which we are dealing with, we provide an overall summary of the data characteristics as follows:

- *g241c* (1 500 points, 241 attributes, 2 classes): This data set has been artificially generated such that the cluster assumption holds, but the manifold assumption does not. The data set consists of 2 Gaussian-distributed classes, each of which with different means and unitary variance. Figure 4.10a shows the PCA projection of this data set on a two-dimensional space.
- *g241d* (1 500 points, 241 attributes, 2 classes): This data set has been artificially generated such that neither the cluster nor manifold assumption holds, but the

smoothness assumption does. This data set contains 2 classes, each of which represented by two Gaussian-distributed classes. The means of each Gaussian are distinct, but the variance is unitary for all 4 Gaussian-distributed classes. Figure 4.10b shows the PCA projection of this data set on a two-dimensional space.

- *Digit1* (1 500 points, 241 attributes, 2 classes): This data set has been artificially generated such that the manifold assumption holds but the cluster assumption does not. It comprises artificial images of the digit “1” with different orientations and distortions. Figure 4.10c shows the PCA projection of this data set on a two-dimensional space.
- *USPS* (1 500 points, 241 attributes, 2 classes): This data set has been designed such as the cluster and manifold assumptions hold. Specifically, it is composed of handwritten digits. There are two classes: digits “2” and “5” are one class and the remainder of the digits is another class. Figure 4.10d shows the PCA projection of this data set on a two-dimensional space.
- *COIL* (1 500 points, 241 attributes, 6 classes): The data in this database only satisfies the manifold assumption. It is made of color images of 100 different objects taken from different angles. Figure 4.10e shows the PCA projection of this data set on a two-dimensional space.
- *BCI* (400 points, 117 attributes, 2 classes): The set of items in this database only satisfies the manifold assumption. It is originated from a research toward the development of a brain computer interface. Specifically, a single person performs a set of trials, in which of each he guesses the movements either with the left or right hands (classes). Figure 4.10f shows the PCA projection of this data set on a two-dimensional space.
- *Text* (1 500 points, 11 960 attributes, 2 classes): The manifold assumptions holds for this data set. It consists of randomly permuted texts. The main task here is separated the word ibm (positive class) from the rest (negative class). Figure 4.10g shows the PCA projection of this data set on a two-dimensional space.

For each aforementioned data set, we consider 10 and 100 labeled vertices. In this way, it is ensured that exists at least 1 labeled vertex for each class in each configuration. For each data set and each quantity of labeled vertices (10 or 100 in these simulations), we generate 12 distinct non-biased label sets. For each of the 12 label sets, the model runs 100 times independently. Finally, the test error of each data set is calculated by averaging these $12 \times 100 = 1200$ runs.

For comparison matters, we have conducted experiments against a selected set of semisupervised techniques, whose simulations results were integrally taken from

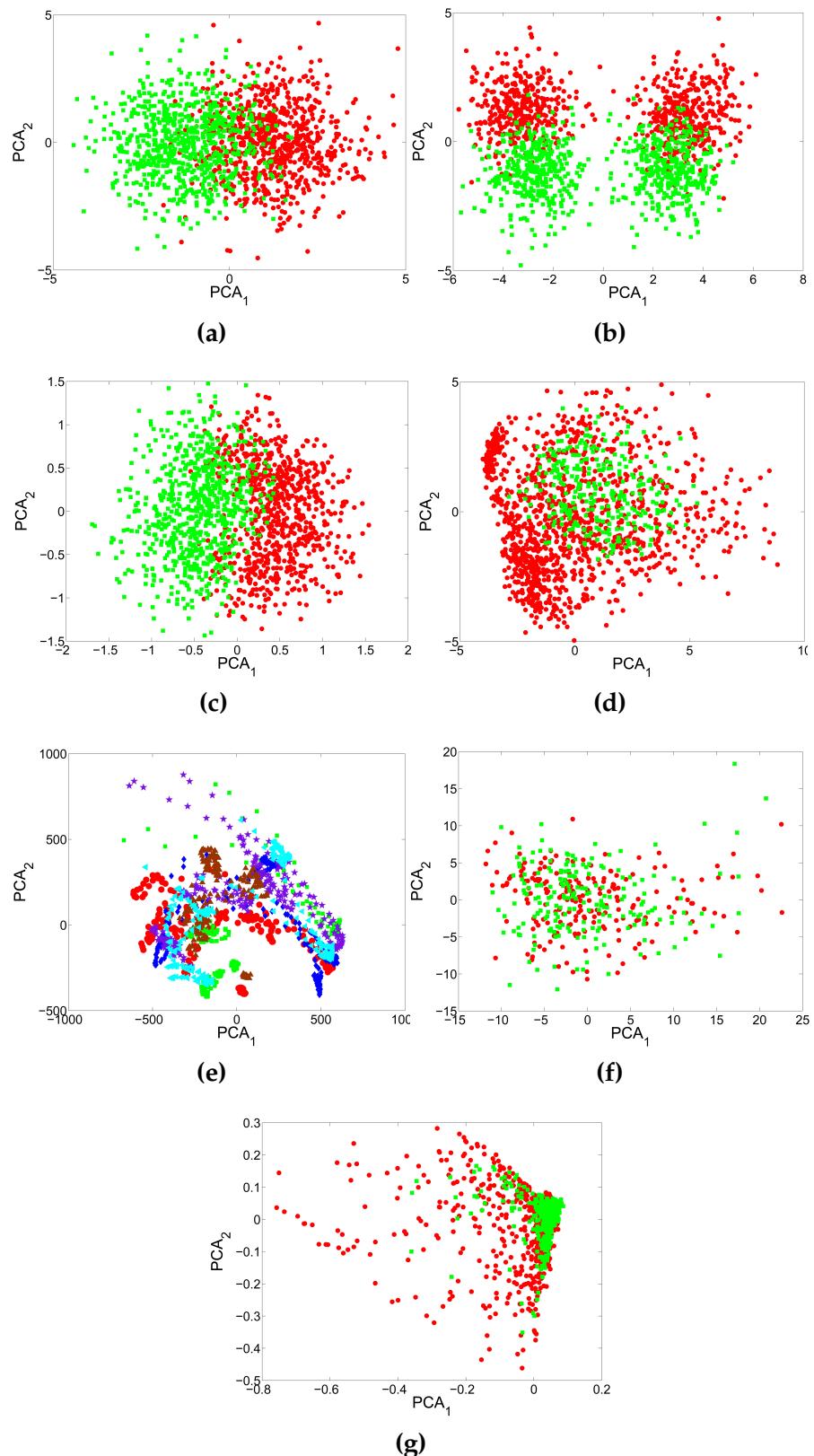


Figure 4.10: PCA projection of the benchmarked data sets. (a) g241c; (b) g241d; (c) Digit1; (d) USPS; (e) COIL; (f) BCI; and (g) Text.

([Chapelle et al., 2006](#)), except for the LGC, LP, and LNP techniques. For the sake of clarity, we supply a summary about each of these techniques in Table 4.3. In brief, the configurations of the competing techniques are ([Chapelle et al., 2006](#)):

- SVM: An RBF kernel is used. For the case where there are 10 labeled instances per partition, the width σ is set to $d/3$, where d is the estimated average distance between every point in the training set with its 10 nearest neighbors.
- MVU + 1-NN: In this spectral method, a number of k nearest neighbors is chosen and the manifold dimensionality is estimated accordingly. k is fixed to 3. After the dimensionality reduction step is completed, an 1-NN classifier to predict the unlabeled vertices is applied.
- LEM + 1-NN: This is another spectral method with the same configuration as the MVU, except for the k parameter, which is fixed at 12.
- QC + CMR: A fully connected graph with an RBF kernel is used. For the case where there are 10 labeled instances per partition, the width σ is set to $d/3$, where d is the estimated average distance between every point in the training set with its 10 nearest neighbors. In relation to the partitions containing 100 labeled instances, the σ is estimated by a cross-validation method on the first split (the other splits use the same σ).
- Discrete Reg.: It is used the energy function with $p = 2$ and $\mu = 0.05$. The graph is constructed using the k -nearest neighbor technique with weights on the edges given by $\exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$. The values of k and γ are selected by a ten-fold cross-validation process over the sets $\{5, 10, 20, 50, \infty\}$ and $\{1/64, 1/16, 1/4, 1, 4, 16, 64\}$, respectively.
- TSVM: An RBF (Radial Basis Function) kernel is used, where its width σ is chosen as the median of the pairwise distances. C is fixed at 100.
- SGT: The hyperparameters are $C = 3200$, $d = 80$, and $k = 100$.
- Cluster-Kernel: The resulting kernel is the product of two other kernels: (i) k_{orig} is a standard RBF and (ii) k_{bag} is the product obtained by running the k -means algorithm repeated times. The k is obtained by a ten-fold cross-validation process.
- Data-Dep. Reg.: Here, it is used k -nearest neighbor regions, centered at each data point as induced by the default Euclidean distance metric. For each region, in order to optimize k , experiments using ten-fold cross-validation processes are run. Data points pertaining to the training set that are disconnected from the graph are treated as errors. The weight of the labeled training data against the unlabeled data (parameter λ in the model) is set to 0.

- *LDS*: ρ is fixed using a standard ten-fold cross-validation process. The other hyperparameters are fixed at their default values according to ([Zeng et al., 2010](#))
- *Laplacian RLS*: The kernel used is of the form: $\tilde{K}(x, z) = K(x, z) - \mathbf{k}_x^T(I + rL^pG)^{-1}L^p\mathbf{k}_z$, where $K(x, z)$ is an RBF kernel with width given by the mean norm of the feature vectors in the data set, $[\mathbf{k}_x]_i = K(x_i, x)$, G is the Gram matrix, L is the normalized graph Laplacian, and r is a constant.
- *CHM (normed)*: This method is used to improve the performance of a supervised base classifier. The simulations are carried out using SVM as its base classifier with a RBF kernel.
- *LGC*: As suggested by the authors in ([Zhou et al., 2004](#)), $\alpha = 0.99$. Moreover, σ is optimized in the interval $0 \leq \sigma \leq 100$.
- *LP*: The parameter σ is optimized in the interval $0 \leq \sigma \leq 100$.
- *LNP*: As suggested by the authors in ([Wang and Zhang, 2008](#)), $\alpha = 0.99$. The parameter k is optimized in the interval $1 \leq k \leq 100$.

Table 4.3: Selected techniques for comparison purposes.

Abbreviation	Technique	Ref(s.)
MVU + 1-NN	Maximum Variance Unfolding	(Sun et al., 2006 ; Weinberger and Saul, 2006)
LEM + 1-NN	Laplacian Eigenmaps	(Belkin and Niyogi, 2003)
QC + CMR	Quadratic Criterion and Class Mass Regularization	(Belkin et al., 2004 ; Delalleau et al., 2005)
Discrete Reg.	Discrete Regularization	(Zhou and Schölkopf, 2006)
TSVM	Transductive Support Vector Machines	(Joachims, 2003 ; Vapnik, 1998)
SGT	Spectral Graph Transducer	(Joachims, 2003)
Cluster-Kernel	Cluster Kernels	(Chapelle et al., 2003)
Data-Dep. Reg.	Data-Dependent Regularization	(Corduneanu and Jaakkola, 2006)
LDS	Low-Density Separation	(Chapelle et al., 2006)
Laplacian RLS	Laplacian Regularized Least Squares	(Sindhwani et al., 2005)
CHM (normed)	Conditional Harmonic Mixing	(Burges and Platt, 2006)
LGC	Local and Global Consistency	(Zhou et al., 2004)
LP	Label Propagation	(Zhu and Ghahramani, 2002)
LNP	Linear Neighborhood Propagation	(Wang and Zhang, 2008)

Regarding the proposed algorithm, we will utilize the following configuration. For comparison purposes, we have used the k -NN graph formation technique with the best value of k in the discretized interval $k \in \{1, 2, \dots, 10\}$ and have also used the best value of Δ in the discretized interval $\lambda \in \{0.2, 0.21, \dots, 0.8\}$. The number of particles inserted

in the model is equal to the number of labeled examples. We set the initial positions of each particle at its representative labeled vertex. The values obtained for the proposed method are averaged by 100 realizations on each of the 12 label configurations. The results obtained from these techniques against the aforementioned databases for 10 initially labeled vertices are reported in Table 4.4, whereas the results for 100 initially labeled vertices are supplied in Table 4.5. In both tables, we have provided the Average Rank of each algorithm, which we have already described in the previous experiment.

A careful analysis of Tables 4.4 and 4.5 shows that our technique had a satisfactory result in comparison to the other techniques. Specifically, for the case of few labeled vertices (10 labeled vertices), our technique reached better results relatively to the other techniques than to the case of 100 initially labeled vertices. This is an attractive characteristic, provided that the task of vertex labeling is often expensive and cumbersome, which generally involves the work of a human expert.

Again, in order to examine the results in a statistical manner, we use an adaptation of the suite of statistical tests presented by (Demšar, 2006) and (Chatfield, 2009) in Tables 4.4 and 4.5. However, since in this case there are some missing results in the table, we will utilize the Skillings-Mack Test, rather than the Friedman Test (it is equal to the Friedman Test when there are no missing values in the table). For our tests here, we fix a significance level of 0.10. According to (Demšar, 2006) and (Chatfield, 2009), we have that $N = 7$ and $k = 17$, resulting in a critical value given by $F(16, 96) \approx 1.55$, where the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. With respect to Table 4.4, we get a value $F_F \approx 1.58$ which is higher than the critical value, so the null-hypothesis is rejected at a 10% significance level. On the other hand, regarding Table 4.5, we get a value $F_F \approx 0.17$, which is not higher than the critical value, hence, we cannot reject the null-hypothesis at a 10% significance level.

As the null hypothesis is rejected for the data presented in Table 4.4 (only 10 labeled examples), we are able to advance to post hoc tests which aim to verify the performance of our algorithm in relation to others by using the Bonferroni-Dunn Test, with the control algorithm fixed as the proposed technique. Thus, if we perform the evaluation of the CD for the problem at hand, we encounter $CD = 4.86$. The average rank of the proposed method is 5.29. By virtue of that, if any rank does lie in the interval 5.29 ± 4.86 , the control and the compared algorithm are statistically the same. Indeed, we conclude that our algorithm is superior to SVM, Discrete Reg., TSVM, and Cluster-Kernel for the set of databases under analysis. However, the other pairwise comparisons to the control do not surpass the CD, meaning that the difference among them are statistically insignificant for the given significance level. Nonetheless, the proposed technique has obtained the best performance (the lowest average rank) in relation to the other techniques for the only 10 labeled examples case.

Table 4.4: Test errors (%) with 10 labeled training points and the corresponding average rank of each technique.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	Avg. Rank
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12	9.86
SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37	14.14
MVU + 1-NN	47.15	45.56	14.42	23.34	62.62	47.95	45.32	9.86
LEM + 1-NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44	10.00
QC + CMR	39.96	46.55	9.80	13.61	59.63	50.36	40.79	7.86
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37	10.86
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21	10.86
SGT	22.76	18.64	8.92	25.36	-	49.59	29.02	6.50
Cluster-Kernel	48.28	42.05	18.73	19.41	67.32	48.31	42.72	10.86
Data-Dep. Reg.	41.25	45.89	12.49	17.96	63.65	50.21	-	9.83
LDS	28.85	50.63	15.63	17.57	61.90	49.27	27.15	8.43
Laplacian RLS	43.95	45.68	5.44	18.99	54.54	48.97	33.68	6.14
CHM (normed)	39.03	43.01	14.86	20.53	-	46.90	-	7.20
LGC	45.82	44.09	9.89	9.03	63.45	47.09	45.50	7.29
LP	42.61	41.93	11.31	14.83	55.82	46.37	49.53	5.57
LNP	47.82	46.24	8.58	17.87	55.50	47.65	41.06	7.43
Proposed Method	43.89	46.47	8.10	15.69	54.18	48.00	34.84	5.29

As our last experiment, we provide a simulation using a large-scale multiclass data set, namely the Letter Recognition Data Set from the UCI Machine Learning Repository. This data set comprises 20 000 samples of the 26 capital letters from the English alphabet, using different fonts and random distortions. The black and white images were converted into 16 primitive numerical attributes (descriptors). We apply two representative graph-based algorithm for comparison matters (LP and LNP using the same parameters as before). Each algorithm is executed using three distinct randomly selected labeled subset sizes, 1%, 5%, and 10%. Table 4.6 reports the test errors for this data set. Again, we see that our method can get good results compared to the others.

4.4 Application: Detection and Prevention of Error Propagation via Competitive Learning

In this section, we will tackle the problem of learning with imperfect data, i.e., some of the labeled samples are incorrectly labeled, via the competitive model described in the previous section.

Table 4.5: Test errors (%) with 100 labeled training points and the corresponding average rank of each technique.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	Avg. Rank
1-NN	43.93	42.45	3.89	5.81	17.35	48.67	30.11	9.00
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45	9.14
MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89	32.83	11.86
LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77	12.14
QC + CMR	22.05	28.20	3.15	6.36	10.03	46.22	25.71	7.50
Discrete Reg.	43.65	41.65	2.77	4.68	9.61	47.67	24.00	8.21
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52	8.71
SGT	17.41	9.11	2.61	6.80	-	45.03	23.09	4.67
Cluster-Kernel	13.49	4.95	3.79	9.68	21.99	35.17	24.38	6.79
Data-Dep. Reg.	20.31	32.82	2.44	5.10	11.46	47.47	-	7.17
LDS	18.04	23.74	3.46	4.96	13.72	43.97	23.15	6.00
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	31.36	23.57	4.93
CHM (normed)	24.82	25.67	3.79	7.65	-	36.03	-	9.10
LGC	41.64	40.08	2.72	3.68	45.55	43.50	46.83	10.00
LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79	9.29
LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.48	12.50
Proposed Method	24.92	29.11	3.11	4.82	10.94	41.57	27.92	7.00

Table 4.6: Test errors (%) obtained for the Letter Recognition data set.

	10% Labeled	5% Labeled	1% Labeled
LP	10.94	18.99	46.94
LNP	24.22	34.08	54.61
Proposed Method	12.09	15.51	38.24

4.4.1 Motivation

The quality of the training data is a fundamental issue in semisupervised learning, because, in this context, less labeled data is available and errors (wrong labels) may easily be propagated to a portion of or the entire data set. Though this is an important topic, it has not received much attention from researchers and there are still few works devoted to the study of semisupervised learning from imperfect data ([Amini and Gallinari, 2003, 2005; Hartono and Hashimoto, 2007](#)). In this section, we treat this important topic - error propagation in semisupervised learning. Usually, in supervised or semisupervised learning, the input label information of the training data set is supposed to be completely reliable. However, in real situations, this is not always true and mislabeled samples are commonly found in the data sets due to instrumental errors, corruption from noise, or even human mistakes in the labeling process. For example, in a medical diagnostic system, the diagnostic results in the training set provided by doctors may be wrong. If these kinds of wrong labels are used to further classify new data (in the supervised learning case) or are propagated to the unlabeled data (in the semisupervised learning), some severe consequences may occur. This situation becomes more critical in autonomous learning, where no external or minimal external intervention is involved. Thus, if the prior knowledge presented to the autonomous learning system contains errors, the performance of the learning system will get worse and worse because of the error propagation. Therefore, considering and designing mechanisms to prevent error propagation is important in the machine learning study and especially in the autonomous learning. Specifically, the prevention of error propagation can benefit the learning systems from two orthogonal aspects:

- i. Improvements of the performance of the learning system, i.e., the system can learn from errors;
- ii. Avoidance of a system's catastrophe by limiting the spreading of wrong labels (input and generated errors).

In this work, we present a new mechanism to prevent error propagation in general semisupervised learning. To our knowledge, many semisupervised learning techniques have been proposed ([Chapelle et al., 2006](#)), but all of them consider that the label information of the labeled subset is totally correct, i.e., there is no error prevention mechanism. In this way, the proposed mechanism together with the analysis and numerical simulations presented in this section make a clear contribution to general machine learning and especially to autonomous learning research.

Specifically, the mechanism for preventing error propagation is designed based on the particle competition and cooperation model previously introduced. This study is important for autonomous learning systems due to the following factors:

1. In many real situations, we have difficulty to get the labels of the whole data set; however, it is easy to have the labels of some of the data items. For example, in a robot soccer game, each robot may not know where is the correct direction to go at each instant, but it does know how to come back when it touches the borderline or how to kick the ball when it is close to it, etc. Such knowledge certainly should be taken into account by the learning system. This means that semisupervised learning is quite a common mechanism in learning systems.
2. In real situations, wrong or conflicting information appears frequently. Thus, a learner cannot be blind-confident on all the information or knowledge that it has at its disposal. Humans and other animals can easily compensate for imperfect data in their learning process. Behavioral experiments show that animals can successfully learn from conditioning even when they are inconsistently rewarded. The ability of treating wrong or conflicting information is essential for autonomous learning systems. One of the ways of treating wrongly labeled data, and consequently preventing their propagation, is here presented.

4.4.2 Detecting Incorrectly Labeled Vertices

In order to detect possible mislabeled vertices, consider the stochastic vector $D(t) = [D^{(1)}(t), \dots, D^{(K)}(t)]$, where the k th-entry, $D^{(k)}(t)$, stores the number of times that particle k has become exhausted until time t . The main idea of introducing this variable is described in the following. Consider the networked data in Fig. 4.11, in which there are two perceivable classes, namely red or dark gray and blue or light gray. The white vertices denote unlabeled vertices, whereas the colored or gray vertices, labeled vertices. Observe that, within the region of the red or dark gray class, there is a mislabeled blue or light gray vertex. In the competitive model, for each labeled vertex, a representative particle is formed. In this way, the vertices in the vicinity of the mislabeled vertices are expected to be in constant competition among the two red or dark gray particles and the single mislabeled blue or light gray particle. Therefore, it is expected that the blue or light gray particle will be stranded in the small region centered at the mislabeled vertex. By virtue of the combination of random and preferential walking of the particle, it will eventually try to venture far away from its represented vertex. Since the surrounding region tends to be heavily dominated by the red or dark gray team, this “false” representative particle will get exhausted very often. Hence, the number of times that a particle becomes exhausted is a good indicator whether the vertex that it is representing is a mislabeled one or not. If the particle is constantly getting exhausted, it is possibly representing a mislabeled vertex. Otherwise, it is probably representing a correctly labeled vertex.

In view of this, the update rule of each entry of $D(t)$ is expressed by:

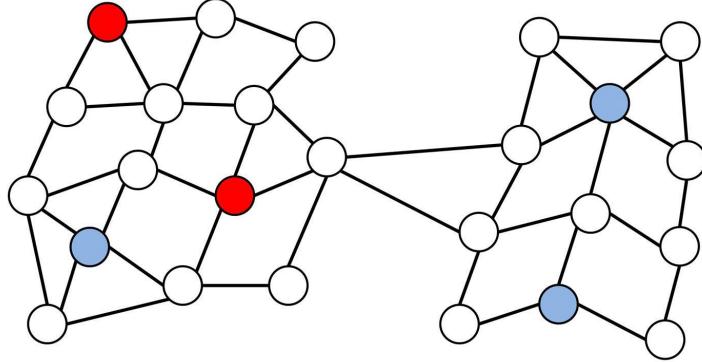


Figure 4.11: A networked data example. Observe that there is a mislabeled blue or light gray vertex within the region that is probably from the red or dark gray class.

$$D^{(k)}(t) = D^{(k)}(t-1) + S^{(k)}(t), \quad (4.20)$$

where $S^{(k)}(t)$ is the boolean-valued variable which indicates whether particle k is active or exhausted at time t . In brief, it yields 1 if particle k is exhausted at time t and 0, otherwise. On account of that, equation (4.20) simply adds 1 or remains with the same summation, depending on the state of particle k at the current time.

A natural question that arises from this scenario is the way one is able to quantify when a particle is getting exhausted more times than the others. Several statistical descriptors could be used for that end, such as the average number of times the particles have become exhausted, or even the median value of the same information. In mathematical terms, these descriptors/thresholds are, respectively, expressed by:

$$\langle D(t) \rangle = \frac{1}{K} \sum_{u=1}^K D^{(u)}(t), \quad (4.21)$$

$$\langle D(t) \rangle = \text{median} \left(D^{(1)}(t), \dots, D^{(K)}(t) \right). \quad (4.22)$$

where $\text{median}(\cdot)$ returns the median value of the provided parameterized list. For simplicity, we assume that the default used descriptor, unless specified otherwise, is the mean value descriptor/threshold described in (4.21). It is worth noticing that other descriptors could be easily used. In view of the statistical stochastic variable introduced in (4.21), any particle k such that:

$$D^{(k)}(t) \geq (1 + \alpha) \langle D(t) \rangle, \quad (4.23)$$

holds is considered to be getting exhausted more times than the other particles. Therefore, such a particle is a great candidate of representing an incorrectly labeled vertex. The parameter $\alpha \in [-1, \infty)$ is a confidence value that indicates the percentage above the average value $\langle D(t) \rangle$ that must occur in order for a particle to be conceived as a representative of an incorrectly labeled vertex. A small α tends to classify more vertices as incorrectly labeled ones than a large value. In the extreme case, when $\alpha \rightarrow \infty$, then the model reduces to its original form, i.e., it does not detect and prevent incorrectly labeled vertices from propagating fake labels.

However, such an approach as it is suffers from a drawback: in the beginning of the dynamical competitive process, a very small portion of the unlabeled data is expected to be dominated by the particles. In this way, the statistical descriptor $\langle D(t) \rangle$ together with the expression displayed in (4.23) may not be correctly describing the real nature of the vertices that each particle represents. Probabilistically speaking, a particle representing a correctly labeled vertex may accidentally get exhausted at the beginning in such a way that (4.23) holds. In an attempt to prevent this false positive, we introduce a weighted function in (4.23), which penalizes when t is small and it yields a constant and nonpenalizing value 1 when t is sufficiently large. The penalization may be understood as a mechanism of preventing the assertion in (4.23) to produce true. A perfect candidate function with these characteristics is a translated exponential decaying function. In this way, the weighted version of (4.23) becomes:

$$(1 - e^{-\frac{t}{\tau}})D^{(k)}(t) \geq (1 + \alpha)\langle D(t) \rangle, \quad (4.24)$$

where $\tau \in (0, \infty)$ is the time constant of the exponential decaying function. Next, we analyze the minimum number of times that a particle must get exhausted in order for (4.24) to hold. For this end, we plot the minimum $D^{(k)}(t)$, denoted as $D_{\min}^{(k)}(t)$, in a such a way that (4.24) holds. Mathematically, it satisfies the following expression:

$$D_{\min}^{(k)}(t) = \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})}. \quad (4.25)$$

For the sake of clarity, let us suppose that the dynamical process produces $\langle D(t) \rangle = 1, \forall t \geq 0$. Fix $\alpha = 0$ for simplicity. In this scenario, Figs. 4.12a and 4.12b plot $D_{\min}^{(k)}(t)$ for which (4.23) and (4.24) hold when $\tau = 25$ and $\tau = 250$, respectively. With respect to the unweighted version, we can observe that even when t is very small, a $D^{(k)}(t) = 1$ is sufficient to (4.23) to be satisfied. Therefore, if a correctly labeled vertex, at the beginning of the competitive process, happens to reach the “exhausted” state, it will be pinpointed as a possible wrongly labeled vertex. On the other hand, the weighted

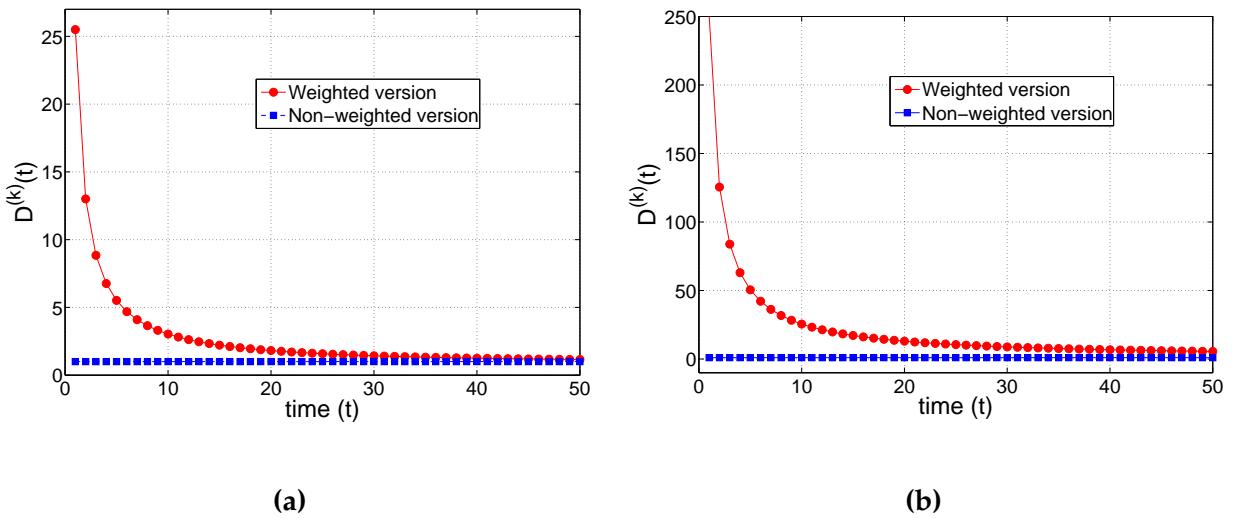


Figure 4.12: Minimum number of times that a particle must get exhausted in order to (4.23) and (4.24) to hold. In the weighted case, the decaying function penalizes $D^{(k)}(t)$ for small t such as to (4.24) never be satisfied. On the other hand, for a large t , the decaying function does not penalize $D^{(k)}(t)$. In the unweighted case, $D^{(k)}(t)$ is never penalized. (a) $\tau = 25$ and (b) $\tau = 250$.

version penalizes $D^{(k)}(t)$ when t is small, as (4.24) reveals. Therefore, it is unlikely that any labeled vertex will be classified as wrongly labeled for a small t . However, for a sufficient large t , this penalization ceases and (4.24) asymptotically approximates (4.23). In particular, when $t \rightarrow \infty$, one has:

$$\begin{aligned} \lim_{t \rightarrow \infty} (1 - e^{-\frac{t}{\tau}}) D^{(k)}(t) &\geq \lim_{t \rightarrow \infty} (1 + \alpha) \langle D(t) \rangle \Rightarrow \\ D^{(k)}(\infty) \lim_{t \rightarrow \infty} (1 - e^{-\frac{t}{\tau}}) &\geq (1 + \alpha) \langle D(\infty) \rangle \Rightarrow \\ D^{(k)}(\infty) &\geq (1 + \alpha) \langle D(\infty) \rangle. \end{aligned} \quad (4.26)$$

i.e., (4.24) reduces to (4.23). Finally, τ is used to control the decaying speed of the exponential function. Looking in isolation to this function, a small τ yields a large negative derivative for this function and a large τ produces a small negative derivative for this function. In other words, the speed of decaying increases as τ decreases.

4.4.3 Preventing the Label Propagation from Incorrectly Labeled Vertices

In the previous section, we have introduced a method for detecting possible wrongly labeled vertices by means of using the information generated by the competitive process itself. Now, once a vertex is pinpointed as a wrongly labeled one, we

need to take actions in order to prevent it from propagating fake labels throughout its neighborhood. For this end, in this section, it is supposed that vertex i has been considered as a possible wrongly labeled vertex at time t , meaning that (4.24) holds.

In view of this, vertex i is going to have its stochastic vector $N_i(t)$ altered, in such a way to reflect how the neighborhood is being dominated at time t . Motivated by the fact that if vertex i is considered as a wrongly labeled vertex, then its neighborhood is probably being dominated by rival particles, we simply restart $N_i(t)$ as the average number of visits received by its neighbors. Mathematically, for all $k \in \mathcal{K}$, we have:

$$N_i^{(k)}(t) = \frac{1}{V_i} \sum_{j=1}^V a_{i,j} N_j^{(k)}(t). \quad (4.27)$$

An important difference from the modified model and the original model is that the former is capable of relabeling labeled vertices, while the latter is not. This new feature is processed when (4.27) is applied.

4.4.4 The New Competitive Learning System

In order to aggregate the detection and prevention of wrongly labeled vertices in the original model, the dynamical system of the original particle competition model presented in Section 4.1 is going to suffer modifications. These are given as follows.

The new internal state of the stochastic dynamical system is given by:

$$X(t) = \begin{bmatrix} p(t) \\ N(t) \\ E(t) \\ S(t) \\ D(t) \end{bmatrix}. \quad (4.28)$$

If $\text{wrong}(k, t) = (1 - e^{-\frac{t}{\tau}})D^{(k)}(t) \geq (1 + \alpha)\langle D(t) \rangle$, then the new competitive dynamical system that supports detection and prevention of incorrectly labeled vertices is given by:

$$\phi : \begin{cases} p^{(k)}(t+1) = j, \quad j \sim \mathbb{P}_{\text{transition}}^{(k)}(t) \\ N_i^{(k)}(t+1) = \mathbb{1}_{[\text{wrong}(k,t)]} \left[\frac{1}{V_i} \sum_{j=1}^V a_{i,j} N_j^{(k)}(t) \right] \\ \quad + \mathbb{1}_{[-\text{wrong}(k,t)]} \left[N_i^{(k)}(t) + \mathbb{1}_{[p^{(k)}(t+1)=i]} \right] \\ E^{(k)}(t+1) = \begin{cases} \min(\omega_{\max}, E^{(k)}(t) + \Delta), & \text{if } \text{owner}(k, t) \\ \max(\omega_{\min}, E^{(k)}(t) - \Delta), & \text{if } \neg \text{owner}(k, t) \end{cases} \\ S^{(k)}(t+1) = \mathbb{1}_{[E^{(k)}(t+1)=\omega_{\min}]} \\ D^{(k)}(t+1) = D^{(k)}(t) + S^{(k)}(t+1) \end{cases} \quad (4.29)$$

It is worth stressing that the modification of the update rule related to the number of visits (2nd expression). Due to the new mechanism of detection and prevention of error propagation, its expression now encompasses two terms: (i) the term which is employed for vertices that have been detected to be wrongly labeled (first term) and (ii) the term for vertices that are not considered wrongly labeled (second term). Specifically, the latter is exactly the update rule of the original system. The logical function $\text{wrong}(k, t)$ is used to check whether the particle k is representing a wrongly labeled vertex. Hence, these two terms are mutually exclusive.

4.4.5 Understanding the Properties and Dynamics of the New Model

In this section, we draw our attention at understanding the properties and dynamics of the model. Specifically, we provide a parameter sensitivity analysis, an investigation of the influences of the initial positions of the particles and the samples, and also how different thresholds impact the detection of incorrectly labeled vertices. It is worth mentioning that, since the proposed technique works in a networked environment, a network formation technique must be employed on vector-based data sets. For this end, we utilize the k -nearest neighbor technique which sets up a link between vertices i and j if i is one of the k nearest neighbors of j or vice versa.

Setting Up the Proposed Algorithm and Environment

In the semisupervised scheme, a set of pre-labeled examples is provided (L is the number of labeled instances). In order to introduce labeled instances with wrong labels (error-prone environment), we deliberately exchange the labels of some pre-labeled instances randomly choosing samples from \mathcal{L} . The proportion of changed instances is denoted as q and the set of mislabeled samples is given by \mathcal{Q} . For example, if $q = 0.1$, then 10% of the labeled instances have their labels flipped, i.e., they turn into incorrectly labeled vertices.

Parameter Sensitivity Analysis

In this section, we will investigate the sensitivity of the parameters used in the detection and the prevention of mislabeled vertices, which are α and τ . Here, we apply the networks generated by the method proposed in (Lancichinetti *et al.*, 2008), which has been explored in Section 2.1.4. The generated networks have $V = 5\,000$ vertices, $\langle k \rangle = 8$, and $\mu = 0.3$. The benchmark process consists in varying the mixing parameter μ and evaluating the resulting accuracy. In order to introduce an error-prone environment, a modification in this benchmark is proposed. Once the network is constructed, we utilize a stratified uniform distribution to compose the labeled set. This labeled set is fixed with the size of 10% of the data set's size. Finally, we deliberately flip 30% of the correct labels to incorrect labels ($q = 0.3$) in a stratified manner, so as to maintain the proportion of labeled samples at each class.

Impact of α

The parameter α is used in the method of signalizing whether or not a vertex is mislabeled (detection process). Equivalently, in the proposed model, this means determining if its corresponding particle is propagating a wrong label. In essence, it determines how much $D^{(k)}(t), k \in \mathcal{K}$, must deviate from $\langle D(t) \rangle$ in order to the corresponding labeled vertex to be conceived as a mislabeled one. Observe that, when $\alpha = -1$, the detection process always accuses that every labeled vertex is possibly mislabeled, because, in accordance with (4.25), one has:

$$D^{(k)}(t) \geq \lim_{\alpha \rightarrow -1} \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})} \implies \quad (4.30)$$

$$D^{(k)}(t) \geq 0. \quad (4.31)$$

Taking into account that the domain of $D^{(k)}(t)$ is the interval $[0, \infty)$, then (4.31) is always satisfied. Therefore, (4.27) is applied to every vertex in the network for any $t \geq 0$. In practical terms, the detection of wrongly labeled data, in this particular case, can be treated as a prefiltering stage in which every (labeled) vertex can preliminarily assess the validity of its label by examining its neighboring (connected) vertices and then spot any inconsistencies. It is worth noting that this kind of strategy is one of the most simple detection schemes in the prevention of error propagation. With this in mind, one can see that the proposed detection method embedded into the particle competition method generalizes this simple strategy in a competitive way. If $\alpha = -1$, the detection procedure reduces to this simple strategy of locally verifying the label validity. Now, when α assumes larger values, the competition dynamics is incorporated to the detection scheme and nonlinear interactions are taken into account in the

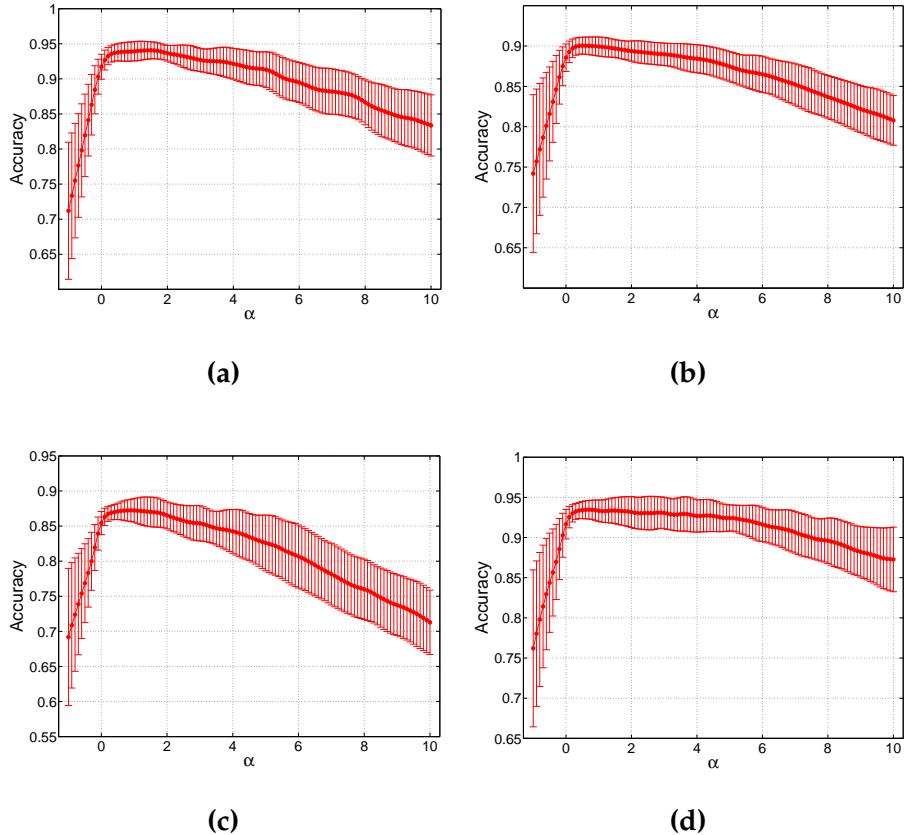


Figure 4.13: Accuracy rate vs. α . We fix $\tau = 40$. Taking into account the steep peak that is verified and the large negative derivatives that surround it, one can see that the parameter α is sensible to the overall model's performance. Results are averaged over 30 simulations. (a) $\gamma = 2$ and $\beta = 1$; (b) $\gamma = 2$ and $\beta = 2$; (c) $\gamma = 3$ and $\beta = 1$; and (d) $\gamma = 3$ and $\beta = 2$.

detection scheme.

As we have drawn attention to, α may take on any values over the interval $[-1, \infty)$. Let us now theoretically analyze the behavior of the detection process for large values of α . As α is increased, one can see that solution space of (4.25) becomes more distant from the origin, meaning that (4.25) is more difficult to be satisfied. One the extreme case, when $\alpha \rightarrow \infty$, one has that:

$$D^{(k)}(t) \geq \lim_{\alpha \rightarrow \infty} \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})} \implies \quad (4.32)$$

$$D^{(k)}(t) \geq \infty, \quad (4.33)$$

that is only satisfied when $t \rightarrow \infty$, which is empirically inviable. Hence, (4.33) never holds for a finite t . As a result, the detection scheme is virtually turned off. In other terms, the particle competition algorithm reduces to its original form proposed in Section 4.1.

Taking into account that analysis, Figs. 4.13a to 4.13d display how the accuracy rate of the model behaves as we vary α from -1 to 10 in the networks constructed using the methodology described in (Lancichinetti *et al.*, 2008) with different values of γ and β . As one can verify from the figures, this parameter is sensible to the outcome of the technique. Oftentimes, the optimal accuracy rate is achieved when a mixture of random and preferential walks occurs. Using a conservative approach, for $0 \leq \alpha \leq 3$, the model gives decent accuracy results when applied to networks with communities.

Impact of τ

The parameter $\tau \in (0, \infty)$ is responsible for adjusting the speed of the penalizing function, used to prevent vertices from being signalized as wrongly labeled ones at the beginning of the stochastic process. Next, we study the behavior of the algorithm for small and large τ in a theoretical and empirical manner.

When τ assumes small values, the exponential decaying function has large-valued derivatives, meaning that its speed of decaying is faster compared to larger values of τ . In the extreme case, i.e., $\tau \rightarrow 0$, one has:

$$D^{(k)}(t) \geq \lim_{\tau \rightarrow 0} \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})} \implies \quad (4.34)$$

$$D^{(k)}(t) \geq (1 + \alpha)\langle D(t) \rangle, \quad (4.35)$$

where we have used the fact that $\lim_{\tau \rightarrow 0}(1 - e^{-\frac{t}{\tau}}) = 1$, since $e^{-\frac{t}{\tau}} \rightarrow 0$ provided that $\tau \rightarrow 0$ and t is finite. This shows that the model's behavior is dictated by the value of α , which was studied in the previous section. This means that, in this special case, the penalizing function ceases to exist, because it decays so fast to be considered relevant in the learning process.

When τ assumes larger values, then the decaying speed of the exponential function reduces accordingly. By virtue of that, we have that:

$$D^{(k)}(t) \geq \lim_{\tau \rightarrow \infty} \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})} \implies \quad (4.36)$$

$$D^{(k)}(t) \geq \frac{(1 + \alpha)\langle D(t) \rangle}{(1 - \lim_{\tau \rightarrow \infty} e^{-\frac{t}{\tau}})} \implies \quad (4.37)$$

$$D^{(k)}(t) \geq \infty, \quad (4.38)$$

because the denominator of (4.38) approaches 0 in a quick manner, since $e^{-\frac{t}{\tau}} \rightarrow 1$ provided that $\tau \rightarrow \infty$ and t is finite. This reveals that the detection scheme is turned off, since there is no reachable solution for (4.38) when t remains finite. Therefore, in

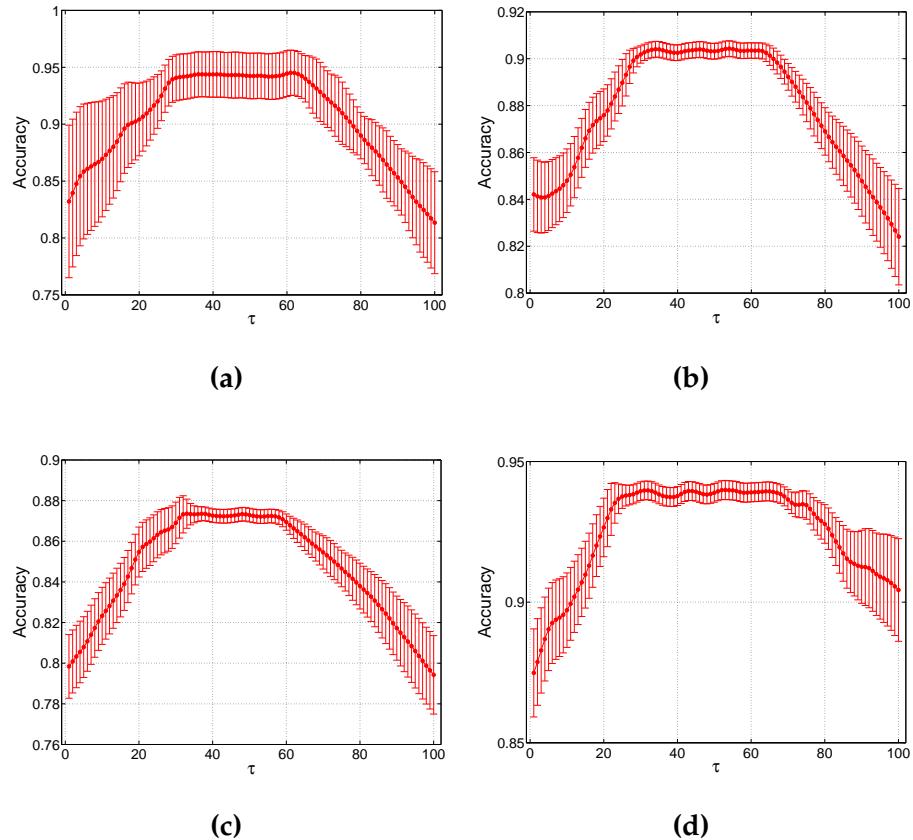


Figure 4.14: Accuracy rate vs. τ . We fix $\alpha = 0$. Taking into account the large steady region that is verified, one can see that the parameter τ is not very sensible to the overall model's performance if one correctly uses it. Results are averaged over 30 simulations. (a) $\gamma = 2$ and $\beta = 1$; (b) $\gamma = 2$ and $\beta = 2$; (c) $\gamma = 3$ and $\beta = 1$; and (d) $\gamma = 3$ and $\beta = 2$.

this case, the model reduces to its original form.

Bearing in mind these considerations, Figs. 4.14a to 4.14d portray the accuracy rate attained by the algorithm for distinct values of τ . We can conclude that, for intermediate values of τ , namely $30 \leq \tau \leq 60$, the model is not sensitive to τ when applied to networks with communities.

Concluding Remarks and Proposed Guidelines for Parameter Selection

We have seen that α is a sensitive parameter and plays a critical role in the detection process, since it dictates when a vertex is considered as a mislabeled one. The computer simulations performed on networks with the presence of communities revealed that the algorithm can supply decent accuracy rates when $\alpha \in [0, 3]$. Moreover, we have verified that τ is not very sensitive to the model's performance, because its accuracy rate remains steady for a considerable interval length. Computer experiments showed that, when $\tau \in [30, 60]$, the model provides reasonable results. Besides that, we have seen that the detection scheme may be turned off depending on the values of α and τ .

Specifically, when $\alpha \rightarrow \infty$ or $\tau \rightarrow \infty$, the detection procedure is turned off. In view of this examination, in all simulations performed on this section, we utilize $\alpha = 0$ and $\tau = 40$. The other model's parameters are utilized according to the suggestions given in Section 4.3.1.

Threshold Sensitivity Analysis

During the model's description, we have discussed about the possibility of utilizing different descriptors/thresholds in the proposed method's detection scheme. For example, (4.21) provides the average value descriptor, while (4.22), the median value descriptor. With this in mind, in this section, we analyze how the accuracy rate of the model behaves when one uses these two kinds of descriptors in artificial networks generated by the already explained methodology detailed in (Lancichinetti *et al.*, 2008). Here, we fix $\alpha = 2$ and $\beta = 2$ with no loss of generality, since, as we have seen in the previous section, the model's accuracy rate behavior is not altered for different values of α and β (see the format of the plots in Figs. 4.13 and 4.14).

Table 4.7 displays how the accuracy rate of the model evolves as we vary the proportion of labeled set (L) and the proportion of wrongly labeled vertices (q). One can see that, when L is small, the average value descriptor has consistently supplied the best results, while, when L is large, the median value descriptor has given the best outcomes. This has happened on account of, when the proportion of the labeled set is small, there is a small quantity of particles walking into the vertices of the networks. In this scenario, the median value descriptor is not a very good option, because it is biased toward the mid-point of the $D^{(k)}(t)$, $k \in \mathcal{K}$. This might not translate well what is really happening in the system, since the representative particles of the mislabeled vertices normally have a much larger value than the representatives of the correctly labeled vertices and also they are the minority. With this descriptor, practically half of these particles would be forced to be considered as mislabeled ones, depending on the value of α (studied in the previous section). On the other extreme, when L is large, the average value may not be a well-behaved selection, because most of the values of $D^{(k)}(t)$ are small while those values corresponding to the mislabeled data are large. In this case, the mean of $D^{(k)}(t)$ ($k = 1 \dots K$) is biased towards the small values. As a result, the detection scheme would accuse several particles of representatives of mislabeled vertices, even if their corresponding $D^{(k)}(t)$ are not very large. Therefore, in this case, a median value would be probably more appropriate.

In view of this analysis, one can see that, when L is small, the average value descriptor is a better choice. But, when L is large, the median might be a better selection. Considering that in real-world data sets the proportion of the labeled set is very small, because the task of manual labeling is cumbersome and often requires the aid of a human expert, then the average value descriptor is recommended under such assumption.

tions.

Table 4.7: Comparison of different thresholds/descriptors with respect to the accuracy rate of the model for varying L and q . Results are averaged over 100 runs.

Accuracy Rate (%)			
	q	Average	Median
L=0.1	0.10	91.82	89.63
	0.25	81.86	86.74
	0.40	81.94	79.01
L=0.3	0.10	94.49	92.97
	0.25	87.52	87.60
	0.40	79.38	77.53
L=0.6	0.10	94.93	95.09
	0.25	86.83	87.40
	0.40	79.04	80.15
L=0.9	0.10	97.07	98.83
	0.25	88.82	90.92
	0.40	79.61	81.74

Influence of the Initial Locations of the Labeled Vertices and Particles

In the competitive network, the trajectories of the particles in the network dictate the performance of the classification and thus the quality of the correction of labels. As a consequence, these trajectories might be different when a mislabeled particle is placed strictly in the region of correctly labeled vertices (normally, in the center of a class) and in the border among classes (frontier between classes). Another important point to be discussed is the influence of the particles' initial positions: does it influence in the overall classification rate of the algorithm? In this section, we will further investigate these two questions using a simple synthetic two-class data set, where each class presents 15 labeled vertices, as illustrated in Fig. 4.15. The black or “star-shaped” vertices denote unlabeled instances, while the red or “circle-shaped” and the blue or “square-shaped” vertices, labeled instances. In order to measure the degree of influence of the initial positions of the vertices composing the labeled set, two different labeling strategies will be used to build up the labeled set, which are:

1. *Random manner*: we randomly draw 15 vertices of each class and label them (stratified uniform distribution), no matter what the characteristics of the selected vertices are (see Fig. 4.15a for an example);
2. *Highest betweenness*: we rank the 15 vertices with the highest betweenness of each class and label them. The betweenness is a centrality measure which is defined

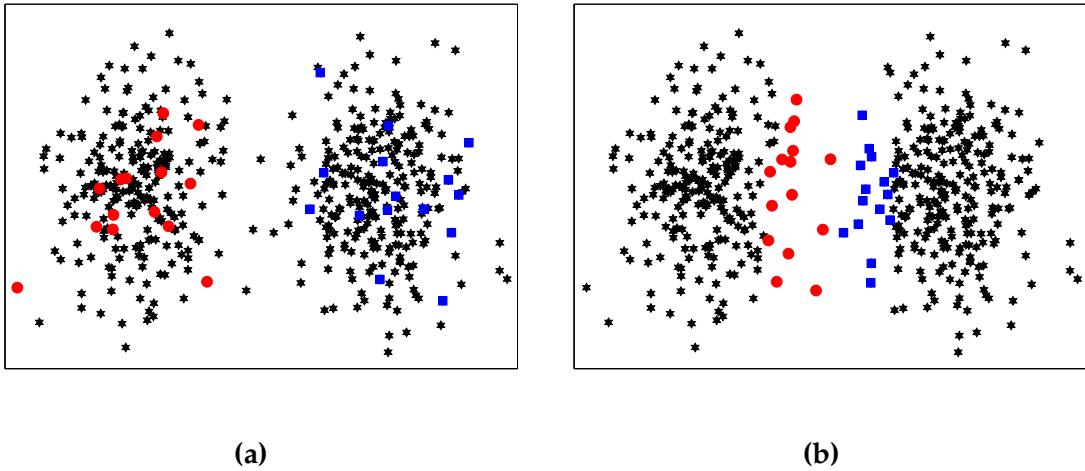


Figure 4.15: Synthetic networks with different strategies for labeling the training set. The k -nearest neighbor with $k = 3$ is employed to build the network (the edges are not shown for the sake of clarity). Labeled sets created using a (a) stratified uniform distribution (15 labeled vertices for each class); (b) using the vertex betweenness centrality measure ([Girvan and Newman, 2002](#)). In this case, we rank the 15 highest vertex betweenness values for each class. Note that all the vertices appear in the frontier from one class to the another.

as the number of geodesic or shortest paths between every pair of vertices in the network that passes by a determined vertex. In this manner, if a vertex usually is contained in several shortest paths, then its betweenness value will be large. In [Girvan and Newman \(2002\)](#), the authors used this measure for community detection tasks, since these vertices are frequently the ones at the *borders* of the class (see Fig. 4.15b for an example).

Table 4.8 reports the accuracy rate of the proposed model when we utilize the synthetic data set depicted in Fig. 4.15 with the two labeling strategies. Within the table, we also provide the simulation results when the particles are spawned at different locations of the network. For example, the entry that goes by the tag [60%, 80%] means that 60% of the red or “circle-shaped” particles are spawned within the red or “circle-shaped” class’ region and the remainder is generated inside the blue or “square-shaped” class’ region. Similarly, 80% of the blue or “square-shaped” particles are put into their corresponding class, while the others 20% are inserted somewhere at the red or “circle-shaped” class. Given the class which the particle is going to be spawned, the vertex is arbitrary chosen using a uniform distribution. A careful investigation of Table 4.8 reveals that, on one hand, our method is not sensitive to the particles’ initial positions, since the accuracy rate does not vary much as one modifies the locations where the particles are spawned. On the other hand, the initial locations of the labeled vertices do influence in the overall accuracy rate of the model because its performance changes when different labeling strategies are utilized.

The reason behind the independence of the model with respect to the particles' initial locations steams from the observation that, with a high probability, a particle will get exhausted and, consequently, restart its walking cycle in its home vertex, as one can see in (4.2). Once it restarts the walking, the model's performance will only depend on where its home vertex is localized. Hence, this also explains the reason that the model is dependent on where the labeled vertices are in the network. In other terms, the particles' initial locations are merely transient to the final dynamics of the competitive system: once they get exhausted, their transient cycles cease and the model's performance will only depend on the initial location of the labeled vertices. This behavior is compelled to happen in view of the reanimation procedure inserted into the dynamics of the model.

A final note that is worth emphasizing is the underlying assumptions that force this transient cycle to be finite. Provided that $\lambda < 1$, then each particle's transition matrix will always have the random walk term not null, as one can verify from (3.1). Considering that the random walk is responsible for the adventurous behavior of the particle, then it will eventually visit a nondominated vertex. In this scenario, probabilistic speaking, for a finite $t > 0$, the particle will get exhausted and the transient cycle will finish. The moment in which this happens depends on the value of the parameter λ : when it is small, the transient cycle tends to be smaller, while, when it is large, it takes a while to finish. Therefore, provided that $\lambda \neq 1$, the transient cycle is guaranteed to complete in a finite t .

Table 4.8: Accuracy rate of the model when different labeling strategies are employed. The particles' initial locations are also reported for the different strategies. Results are averaged over 100 runs.

Accuracy [%]	Strategy	Proportion of Particles in Each Class [%]						
		[100, 100]	[80, 80]	[60, 60]	[40, 40]	[20, 20]	[0, 0]	[100, 0]
$q = 0$	Random Betweenness	98.93	98.90	98.96	98.88	99.01	98.98	99.00
		81.22	81.30	81.34	81.20	81.19	81.23	81.21
$q = 0.1$	Random Betweenness	96.49	96.51	96.45	96.43	96.55	96.49	96.48
		76.75	76.71	76.74	76.75	76.77	76.74	76.72
$q = 0.2$	Random Betweenness	91.02	91.04	91.01	91.01	91.00	91.04	91.03
		70.48	70.48	70.51	70.50	70.49	70.48	70.50
$q = 0.3$	Random Betweenness	87.22	87.17	87.25	87.26	87.23	87.20	87.21
		64.81	64.76	64.79	64.76	64.80	64.83	64.83
$q = 0.4$	Random Betweenness	80.36	80.45	80.41	80.37	80.32	80.36	80.44
		57.15	57.21	57.23	57.25	57.19	57.24	57.26

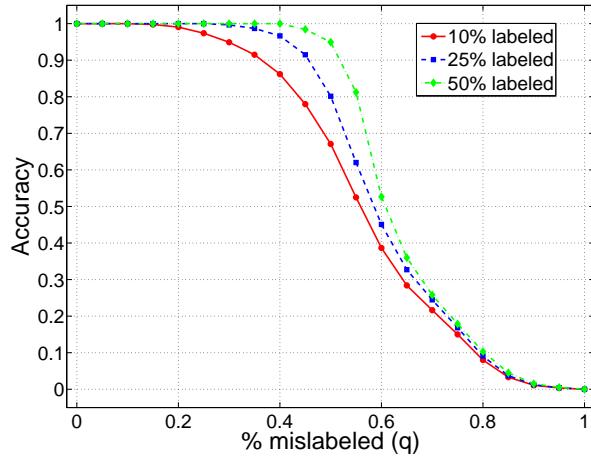


Figure 4.16: Behavior of the accuracy rate of the model vs. the proportion of mislabeled instances (q), when random clustered networks with constant mixture are used. The accuracy rate behavior is displayed for three different proportions of labeled instances. The networks' configurations are: $V = 10\,000$, $M = 16$, and $z_{\text{out}}/\langle k \rangle = 0.3$. 100 independent runs are performed and the average value is reported.

4.4.6 Computer Simulations

In the next sections, computer simulations are performed with the goal of quantifying the robustness of the modified version of the particle competition model (PCM) in a semisupervised error-prone environment.

Computer Simulations on Synthetic Data Sets

In this section, we will study the behavior of the proposed algorithm on artificial networks. Specifically, we will make use of Girvan-Newman's benchmark, whose construction method has already been described in Section 2.1.4. Figure 4.16 shows the behavior of the accuracy rate of the model against the proportion of mislabeled instances (q) for three different sizes of the labeled set. We can clearly spot three different regions, separated by two critical points of interest, as follows:

- When q is small, the effect of the mislabeled vertices on the accuracy rate of the algorithm is minimal. Visually, this is translated by the plateau region in the plot depicted in Fig. 4.16 with basis near the 100% accuracy rate. This behavior can be explained by the competition that happens in the dynamical system as it evolves in time. As the grand majority of the labeled samples is correctly labeled, the competition among the wrongly and correctly labeled will greatly favor the correctly labeled teams. In other words, the propagation of the correctly labeled samples will literally overwhelm the propagation originated by the mislabeled samples. As a consequence, the performance of the algorithm is slightly altered.

- When q is intermediate, the accuracy rate largely drops at a considerable rate. This marks the point (first critical point) in which the wrongly labeled vertices' propagation starts to overwhelm the propagation of the correctly labeled ones. Since the network is heterogeneous, the locations of the initial labeled samples play an important role in characterizing when this sudden decay will begin taking scene. This empirical behavior confirms our investigation undertaken in the previous section. Moreover, one can see that this phenomenon is also dependent on the size of the labeled set. As the labeled set gets bigger and more representative, the more robust the algorithm will be in an error-prone environment. Visually, this can be inspected in the plot by the narrower drop-off region inbetween these two critical points. For instance, when 50% of the data set are labeled, this region is smaller than the one yielded when 10% of the data set are labeled.
- When q is large enough, the accuracy rate reaches another steady threshold again, but now placed at a low accuracy rate. The beginning of this phenomenon characterizes the second critical point. At this point, the propagation of wrongly labeled vertices by misrepresented particles completely overwhelms the particles that are spreading correctly labeled samples. The region beyond the second critical point is of little relevance for us, since the quality of the labeled set is worse than a mere random labeling scheme. In these special situations, one could use unsupervised learning tasks and expect much better results, because they do not use the external and misinterpreted labeling information in their learning process.

It is worth emphasizing that the locations of the first and second critical points provide an important indicator of the robustness of the algorithm in an error-prone environment. Specifically, as the distance between these points reduces, the higher is the robustness of the algorithm. With that in mind, in our experiments, the proposed method is considerably more robust as the size of the labeled set grows. This is expected, since the labeled set becomes more representative and, hence, hard to be incorrectly discovered by the competition process among the particles.

With that observation in mind, in the following computer simulation, we inspect the occurrence of the first critical point as the network mixture $z_{\text{out}}/\langle k \rangle$ grows. First, we provide the procedure used to detect this critical point as follows. From Fig. 4.16, we can verify that there are 2 inflection points, which turn out to be the critical points that we are looking for. In order to easily spot them, given the function which maps the proportion of mislabeled samples to an accuracy rate, we take the second derivative of it. The points in which this second derivative is null mark the inflection points. The first critical point is the inflection point with smaller magnitude. The function which maps q to an accuracy rate is constructed using the proposed algorithm with a labeled set constituted by 10% of the samples in the data set. Figure 4.17 depicts the

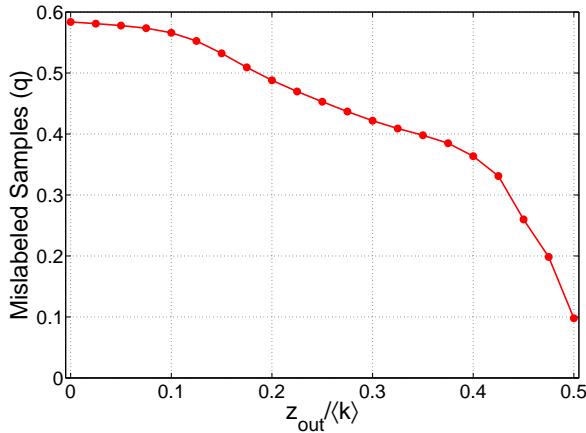


Figure 4.17: First critical point vs. network mixture $z_{\text{out}}/\langle k \rangle$. We label 10% of the samples in the data set. The networks' configurations are: $V = 10\,000$, $M = 16$. 100 independent runs are performed and the average value is reported.

first encountered critical points against different network mixtures. One can see that, as the network mixture grows, the communities start to get harder to be discovered. As a consequence, the first critical points are already verified with small proportions of mislabeled samples (q). Finally, it is valuable to say that we have stopped the network mixture axis at 0.50 (rather than 1), because after this point, the communities are no longer defined in the stronger sense.

Computer Simulations on Real-World Data Sets

In this section, the proposed algorithm is applied to real-world data in an imperfect labeled data environment. For comparison matters, a set of competing semisupervised learning techniques is also employed, which is summarized in Table 4.9. With respect to the model's selection, all the parameters are tuned in accordance with the best accuracy rate reached the algorithms. The model selection is conducted as follows:

- *D-Walks*: L is selected over the discretized interval $\mathcal{L} \in \{1, \dots, 10\}$ (as suggested by (Callut *et al.*, 2008, Fig. 2));
- *LP*: σ is selected over the discretized interval $\sigma \in \{0, 1, \dots, 100\}$ and α is fixed to $\alpha = 0.99$ (the same setup as (Zhou *et al.*, 2004));
- *LNP*: k is evaluated over the discretized interval $k \in \{1, 2, \dots, 100\}$, and σ , as well as α , are selected using the same process employed in the LP parameters selection;
- *SS Modularity*: this is a nonparametric technique.

Table 4.9: Description of some semisupervised data classification techniques used in the error propagation analysis.

Abbreviation	Technique	Reference
D-Walks	Discriminative Walks	(Callut <i>et al.</i> , 2008)
LP	Linear Propagation	(Zhou <i>et al.</i> , 2004)
LNP	Linear Neighborhood Propagation	(Wang and Zhang, 2008)
SS Modularity	Semisupervised Modularity	(Silva and Zhao, 2012c)
Original PCM	Original Particle Competition Method	Section 4.1

As for the original and proposed PCMs, the data are firstly transformed into a network representation using the k -nearest neighbor graph formation technique. For this purpose, k is chosen over the discretized interval $k \in \{1, 2, \dots, 10\}$. For the model's parameter, we first discretize the interval of λ by using a step equal to 0.02, i.e., $\lambda \in \{0.20, 0.22, \dots, 0.80\}$, as suggested by the guidelines that we have seen. Then, the proposed technique is run against an input data set by using each of the combinations of the discretized values of λ and k . The result which produces the best classification accuracy is selected. With respect to the modified version of the PCM, the parameter selection is processed in accordance with the analysis supplied in Section 4.4.5.

We apply the LP, LNP, the original PCM and the modified PCM on two data sets from the UCI Machine Learning Repository (Frank and Asuncion, 2010): Iris and Letter Recognition. The former is composed of three equal-sized classes, each of which comprising 50 samples, totalizing 150 samples. The latter is composed of 20 000 samples divided into 26 unbalanced classes, each representing a different letter of the English alphabet. Thus, the Letter Recognition data set can be considered as a large-scale data set.

Figures 4.18a and 4.18b show the behavior of the test error vs. the proportion of mislabeled samples q . One can verify that, as q grows, all algorithms start to produce larger test errors. However, the proposed technique (proposed PCM) is able to outperform all the compared algorithms, by virtue of the detection and prevention mechanisms embedded into the competitive model. One point that is worth mentioning is, when the environment is error-free ($q = 0$), then the proposed and original PCMs supply very similar results. This is expected, since the labeled samples are hoped to lie within densely connected groups; therefore, the owner switching mechanism introduced in (4.25) will rarely be satisfied. The proposed PCM is able to maintain a decent accuracy rate because of the competitive mechanism realized by the particles. In an error-prone environment, we can conceive the algorithm as having two types of competition taking place simultaneously: competition of particles spreading correctly and incorrectly labeled samples. Since the competition is always taking place indirectly (through the accumulated domination levels of each vertex and particles' movement policy), then

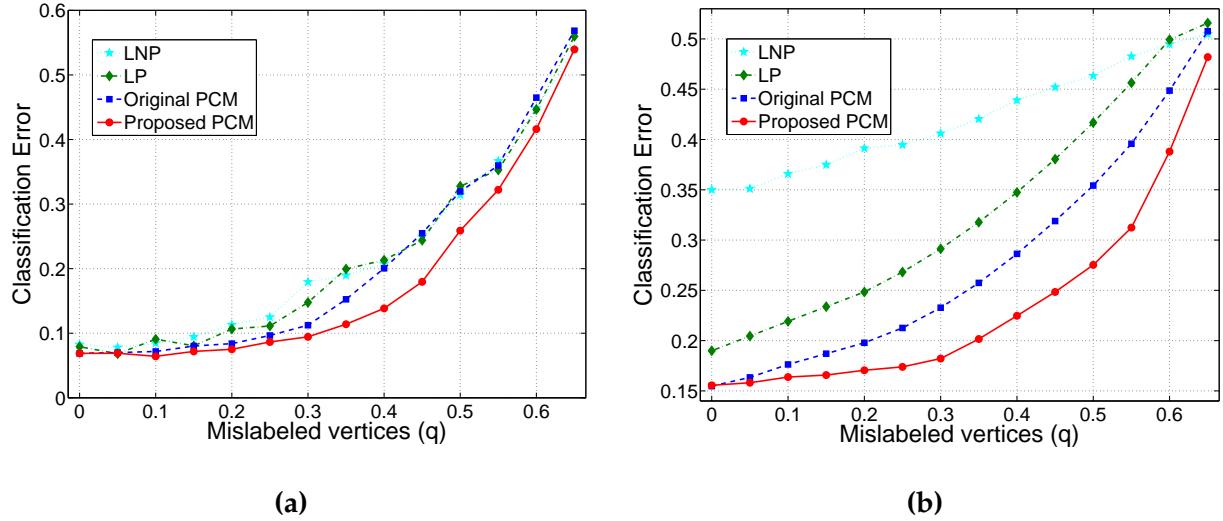


Figure 4.18: Behavior of the classification error as the proportion of wrongly labeled vertices increases on two real-world data sets. (a) Iris data set and (b) Letter Recognition Data Set. 100 independent runs are performed and the average value is reported.

these two types of label spreading are always in confront. As it is expected to one have more correctly labeled samples than incorrectly ones, then the competition will naturally extinct the propagation of the mislabeled samples as the system progresses in time.

Next, the benchmark proposed in (Chapelle *et al.*, 2006) will be used, which we have already described in the previous sections of this chapter. The test error results, when there are only 10 and 100 initially labeled vertices, are reported in Table 4.10 and 4.11, respectively, for 4 different proportions of mislabeled samples: 0, 0.1, 0.2, and 0.4. One can see that, as the proportion of mislabeled samples grows, all the 3 competing techniques start to yield high test error rates. However, the proposed technique is able to perform in a decent manner even in an environment with 40% of mislabeled vertices. The reason behind this is that competition is an excellent way of vanishing the mislabeled vertices, given that the majority of the labeled samples is correctly labeled. As stated before, the team of correctly labeled vertices completely overwhelms the mislabeled propagators, i.e., the mislabeled particles, as the dynamical system evolves in time.

For our statistical tests, we apply the already discussed suite of statistical tests indicated by (Demšar, 2006). A significance level of $\alpha = 5\%$ is fixed. Applying the first step, i.e., the Friedman test, we can conclude that the ranks of the set of algorithms presented in Tables 4.10 and 4.11 are statistically significant from the null hypothesis, which states that the rank of the algorithms are the same. Since a statistical difference has been spotted, we are able to apply the second step using the Bonferroni-Dunn test. In such test, we compare the performance of the proposed algorithm (control algo-

Table 4.10: Test error (%) obtained by the four techniques under comparison on clean and noisy data sets extracted from the Chapelle's benchmark. The labeled set contains only $L = 10$ labeled vertices.

	Technique	g241c	g241d	Digit1	USPS	COIL	BCI	Text
$q=0$	D-Walks	45.41	46.99	8.01	19.63	54.28	47.85	39.17
	LP	42.61	41.93	11.31	14.83	55.82	46.37	49.53
	LNP	47.82	46.24	8.58	17.87	55.50	47.65	41.06
	SS Modularity	44.82	46.92	8.27	16.83	55.30	48.05	36.66
	Original PCM	43.89	46.47	8.10	15.69	54.18	48.00	34.84
	Proposed PCM	43.53	46.41	8.06	15.71	54.07	47.88	34.70
$q=0.1$	D-Walks	44.92	43.92	15.72	17.22	57.62	48.97	43.78
	LP	44.77	44.01	17.85	19.82	58.93	47.50	49.82
	LNP	48.93	49.04	20.33	22.75	61.51	48.98	44.00
	SS Modularity	46.88	45.01	11.73	20.96	59.08	50.25	47.99
	Original PCM	45.27	48.05	12.10	19.99	58.44	48.91	38.12
	Proposed PCM	43.96	47.08	9.34	15.27	54.55	48.19	36.01
$q=0.2$	D-Walks	46.94	49.91	18.27	27.88	60.72	49.27	40.22
	LP	45.11	47.28	23.20	25.44	59.72	48.39	50.74
	LNP	50.53	49.79	23.24	26.86	64.90	49.25	46.67
	SS Modularity	46.83	50.00	18.51	26.65	61.47	49.21	40.63
	Original PCM	47.10	48.88	16.25	24.70	61.34	49.54	40.03
	Proposed PCM	44.76	48.00	11.07	16.73	54.87	49.11	38.45
$q=0.4$	D-Walks	55.82	61.83	39.95	40.38	75.01	54.70	56.08
	LP	57.83	60.16	41.80	49.01	76.39	55.27	58.33
	LNP	58.06	59.63	46.73	47.50	81.35	55.47	56.88
	SS Modularity	54.83	60.99	39.53	41.06	75.58	54.14	56.40
	Original PCM	52.95	60.04	38.33	39.71	73.64	53.09	56.42
	Proposed PCM	47.83	49.15	24.99	29.84	64.30	52.00	44.28

Table 4.11: Test error (%) obtained by the four techniques under comparison on clean and noisy data sets extracted from the Chapelle's benchmark. The labeled set contains only $L = 100$ labeled vertices.

	Technique	g241c	g241d	Digit1	USPS	COIL	BCI	Text
$q=0$	D-Walks	23.95	30.03	3.17	4.66	11.39	42.00	27.55
	LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79
	LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.48
	SS Modularity	23.99	30.71	3.02	5.14	10.11	43.24	27.90
	Original PCM	24.92	29.11	3.11	4.82	10.94	41.57	27.92
	Proposed PCM	24.90	29.03	3.10	4.71	10.89	41.54	27.86
$q=0.1$	D-Walks	25.48	30.38	4.38	6.02	13.00	44.31	28.64
	LP	31.59	30.45	4.77	8.02	13.41	43.66	41.90
	LNP	45.42	40.00	4.64	19.98	12.93	46.70	39.65
	SS Modularity	25.97	31.06	4.02	6.13	12.72	45.02	29.32
	Original PCM	25.39	30.55	4.20	6.06	12.17	42.78	28.99
	Proposed PCM	24.94	29.27	3.13	4.82	11.05	41.57	28.20
$q=0.2$	D-Walks	29.94	34.09	10.11	9.09	21.64	50.35	32.96
	LP	34.48	37.31	14.76	16.04	20.27	45.93	45.01
	LNP	36.27	39.05	21.53	25.50	34.85	50.27	45.59
	SS Modularity	31.83	38.63	17.48	11.38	29.81	52.00	39.42
	Original PCM	28.73	32.19	7.77	8.15	18.00	46.11	31.73
	Proposed PCM	26.37	30.61	5.22	5.40	11.39	42.01	29.14
$q=0.4$	D-Walks	49.66	53.84	48.73	34.72	40.05	56.60	41.28
	LP	43.26	47.90	30.31	39.19	42.65	50.20	48.51
	LNP	51.58	52.20	40.49	42.34	51.46	54.21	53.00
	SS Modularity	50.37	51.11	42.26	38.29	49.87	60.81	51.29
	Original PCM	39.12	42.02	27.34	28.27	35.70	48.63	39.85
	Proposed PCM	29.85	33.96	9.73	10.10	16.88	44.37	32.09

rithm) with regard to each of the competing techniques. This test provides, for each pair of techniques, whether they are statistically significant or not. The null hypothesis here is that the performance of the pair of algorithms is the same. The pairwise comparisons are given in Table 4.12. One can see that, as q gets larger, the error detection and prevention of the modified PCM are responsible for maintaining the decent accuracy rate. On the contrary, the other techniques, as there is no such mechanism, suffer from the incorrectly labeled vertices and their accuracy rate drops accordingly. We conclude that the proposed detection and prevention procedure are able to maintain good results in real-world data sets, since the accuracy rate difference from $q = 0$ and $q = 0.4$ is far less from the other techniques. A last note goes to the original and modified PCM algorithms, which are able to withstand environments of considerable noises. This happens because the competition mechanism is a perfect candidate for vanishing mislabeled vertices by utilizing the local neighborhood in the learning process. Since the original PCM does not admit flipping of already labeled vertices, the mislabeled vertices continue to propagate their labels, but in a very restrictive area, where the neighborhood is strongly dominated by other teams of particles. However, in the modified version, once the learning process detects a possible mislabeled vertex, it flips its label, enabling and explaining the higher accuracy rates reached in Tables 4.10 and 4.11.

Table 4.12: Analysis of whether the modified PCM is statistically superior to the competing techniques when the Bonferroni-Dunn Test is applied. These results apply to Tables 4.10 and 4.11 (10 and 100 initially labeled vertices, respectively).

q	LP	LNP	SS Modularity	D-Walks	Original PCM
0	Yes	Yes	No	No	No
0.1	Yes	Yes	No	No	No
0.2	Yes	Yes	Yes	No	No
0.3	Yes	Yes	Yes	Yes	No
0.4	Yes	Yes	Yes	Yes	Yes

4.5 Chapter Remarks

This chapter proposes a new semisupervised learning technique using a particle competitive-cooperative mechanism. In this model, several particles, each of which representing a class, navigate in the network to explore their territory and, at the same time, attempt to defend their territory against rival particles. If several particles propagate the same class label, then a team is formed, and a cooperation process amongst these particles occurs.

In this chapter, we also propose a method for detecting and preventing error propagation embedded in the semisupervised learning technique. The error detection mechanism is realized by weighting the total number of times a particle has become exhausted to a thresholded value, which is dependent and vary in time. When the dynamical competitive system begins, there is a penalizing factor which prevents the detection of false positives. This has been introduced in order to diminish the dependency of the proposed error propagation model on the initial locations of the labeled samples (transient part of the dynamics). As the system evolves, this penalization ceases to exist and the plain domination level that each vertex has is used in the error propagation inference. Once a vertex is declared as mislabeled, the proposed technique resets its domination levels as the average value of its neighborhood, so as to conform to the cluster assumption that the proposed algorithm holds on to.

Computer simulations have been conducted and satisfactory results have been obtained. Specifically, we have shown that, as the proportion of mislabeled samples increases, the proposed method is still able to yield decent accuracy rates by virtue of the aforementioned mechanism. In addition, we have investigated how the first critical points of the model relate to the network mixture on random clustered networks. Finally, we have adapted the benchmark proposed in ([Chapelle et al., 2006](#)) to an error-prone environment. In this modified version, we have tested the proposed algorithm against state-of-art techniques. In the majority of the cases, the proposed method is able to outperform them when a large portion of samples is mislabeled.

As many semisupervised learning techniques consider label propagation, we here treat error propagation. This is a fundamental question in machine learning because mislabeled samples are commonly found in the data sets due to several factors, such as instrumental errors, corruption from noise, or even human mistakes. In autonomous learning systems, errors are much easier to be propagated to the whole data set due to the absence or few external intervention, which makes the situation more critical. This work is an endower to this direction.

As a future work, it is possible to develop a new learning technique by combining the active learning and semisupervised learning approaches. Specifically, we may use active learning to select data instances to be labeled and then we use semisupervised learning to really generate labels for the remainder of the data items. Due to the selection process performed by the active learning, we believe the combination of both types of learning is more resistant to errors presented in the training set.

CHAPTER
5

Supervised High Level Data Classification in Complex Networks

This chapter treats the issue of supervised data classification by using not only physical features of the data items, but also high level characteristics of them. It worth reiterating that most methods in the literature ignore the high level relations among the data, such as the formation of clear patterns in the data. In view of this gap, we present a hybrid classifier that takes into account both types of learning. Roughly speaking, the low level classifier may be implemented by any traditional techniques. On the other hand, the high level classification exploits the complex topological properties of the underlying network constructed from the input data. In the proposed framework, the two classifiers are joined together via a linear convex combination, which is calibrated by the *compliance term*. In the developed work, two different kinds of high level classifiers are provided, both relying on a networked representation of the data, as follows:

- The first one comprises a weighted combination of three network measures, namely the assortativity, the clustering coefficient, and the average degree;
- The second one is composed of two quantities that are directly derived from the dynamics of a tourist walker, which are the cycle and the transient lengths.

Studies are going to be provided in order to show the impact of different compliance terms for different types of data sets, ranging from completely well-posed classes to highly overlapping classes. As a quick glimpse of our final results, we will see that one must raise the influence of the decision of the high level classifier as the classes' configurations become more complex.

Once the technique is properly presented, we will further explore the effectiveness of the model by delving into the real-world application of handwritten digits and letters recognition, where we show that the high level term can really improve the accuracy of the model. Additionally, an illustrative network composed of manuscript digits is going to be investigated, where we try to show the power that the compliance term makes upon the final decision of the classifier.

5.1 Motivation

Oftentimes, the data items are not isolated points in the attribute space, but instead tend to form certain patterns. For example, in Fig. 5.1, the test instance represented by the “triangle” (black) is most probably to be classified as a member of the “square” (blue) class if only physical features, such as distances among data instances, are considered. On the other hand, if we take into account the relationship and semantic meaning among the data items, we would intuitively classify the “triangle” item as a member of the “circle” (red) class, since a clear pattern of a “house” contour is formed. The human (animal) brain performs both low and high orders of learning and it has facility in identifying patterns according to the semantic meaning of the input data. However, this kind of task, in general, is still cumbersome to be assessed by computers. Supervised data classification which not only considers physical attributes but also pattern formation is here referred to as *high level classification*.

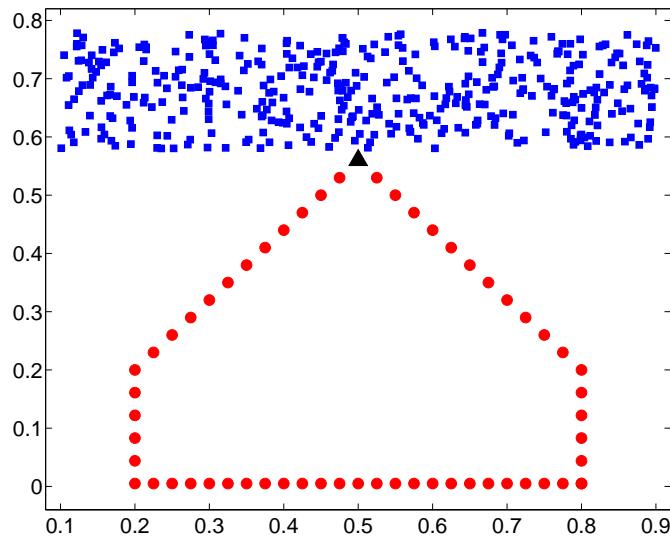


Figure 5.1: A simple example of a supervised data classification task where there exists a class with a clear pattern, in this case, the red (“circle”) class. The goal is to classify the black “triangle” data item. Traditional (low level) classifiers would have trouble in classifying such item, since they derive their decisions merely based on physical measures.

There are several kinds of works related to high level classification. One of them is the co-training technique ([Blum and Mitchell, 1998](#)), which is applied to semisupervised learning to process data sets with labeled and unlabeled instances. It requires two views of the data and it assumes that each example is described using two different feature sets that provide different, complementary information about the instances. Co-training first learns a separate classifier for each view using the labeled examples. The most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data. By considering different views of the same data set, some kinds of data relationships determined by the predefined views may be uncovered. Another related technique is the committee machine, which consists of an ensemble of classifiers ([Haykin, 2008](#)). In this case, each classifier makes a decision by itself and all these decisions are combined into a single response by a voting scheme. The combined response of the committee machine is supposed to be superior to those of its constituent experts. Since each classifier has a particular view with respect to the input data, the combination of them may reveal relationships among input data.

The manifold learning ([Lee and Verleysen, 2007](#); [Seung and Lee, 2000](#); [Tenenbaum et al., 2000](#)) is also related to this work. It assumes that the data of interest lies on an embedded nonlinear manifold within a higher dimensional space. Data transformations, for example, Isomap, Locally Linear Embedding, etc., are designed to reduce the dimension of the data while preserving the local linear relationships between neighbors. However, as we will show in this chapter, manifold learning cannot correctly identify some kinds of data patterns. For example, in Fig. [5.8a](#), any red or “circle” data point passing through the green or “square” class will not be correctly classified as an element of the red or “circle” class, because each of them has many more green or “square” neighbors than red or “circle” neighbors.

Another strongly related work is the Semantic Web ([Feigenbaum et al., 2007](#); [Shadbolt et al., 2006](#); [Tim Berners-Lee and and Lassila, 2001](#)). The idea of Semantic Web is “an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” ([Tim Berners-Lee and and Lassila, 2001](#)). Semantic Web uses ontologies to describe the semantics of the data. Even though it is a promising idea, it still presents several difficult challenges. A key challenge in creating Semantic Web is the semantic mapping among the ontologies, i.e., there are more than one ontology to describe the same data item. The difficulty resides in encountering a map of ontologies that are semantically related. Another challenge is the one-to-many mapping of concepts in the Semantic Web, since most techniques are only able to induce one-to-one mapping, which does not correspond to real-world problems.

Statistical relational learning, specifically statistical relational classification (SRC),

is another strongly associated topic. It predicts the category of an object based on its attributes and its links and attributes of linked objects. According to Ref. (Gallagher *et al.*, 2008), graph-based SRC is a within-network classification task and it can be categorized into two main groups: collective inference (Gallagher *et al.*, 2008; Macskassy and Provost, 2007; Skolidis and Sanguinetti, 2011; Zhang and Mao, 2008; Zhang *et al.*, 2011, 2006; Zhu *et al.*, 2007) and graph-based semisupervised learning (Chapelle *et al.*, 2006; Silva and Zhao, 2012a,b,c; Zhu, 2005a). In both cases, the labels are propagated from pre-labeled vertices to unlabeled vertices considering the local relationships or certain smoothness criteria. On the other hand, the classification method introduced here presents a different approach. It does not consider class label propagation within the network, instead, it characterizes the pattern formation of the whole network by exploiting its topological properties. The classification is made by checking the compliance of each input data with the pattern formation of the underlying network.

Maybe the most related works to high level classification are the contextual classification techniques (Binaghi *et al.*, 2003; Lu and Weng, 2007; Micheli, 2009; Tian *et al.*, 2000, 2006; Tuia *et al.*, 2010; Williams *et al.*, 2007), which consider the spatial relationships between the individual pixels and the local and global configurations of neighboring pixels for assigning classes. The underlying assumption is that the neighboring pixels are strongly related, i.e., structures of classes are likely to be found in the presence of others. In the proposed high level classification method, the spatial information is only used in the network construction phase. The difference between the contextual classification and the proposed approach lies in the following factor: once the network is constructed from the original data, the proposed classification method does not use the spatial relation among data items anymore, but instead it exploits the topological properties of the network in search of pattern formation. The topological properties have been shown to be quite useful in data analysis. For example, they are applied to detect irregular cluster forms by a data clustering or a semisupervised classification algorithm with a unique distance measure (Karypis *et al.*, 1999).

From the viewpoint of general high level classification, all the abovementioned approaches are quite restricted either to the types of semantic features to be extracted, such as co-training and ensemble method, or to the types of data, such as the contextual classification, which is devoted to considering spatial relationship among pixels in image processing. To our knowledge, it is still lacking an explicit and general scheme to deal with high level classification in the literature, which is quite desirable for many applications, such as invariant pattern recognition. The current work presents an endower to this direction.

In this chapter, a hybrid classification technique which combines the low and high levels of learning is going to be presented. Generally speaking, the low level classification can be implemented by any traditional classification technique. On the other

hand, the high level classification exploits the complex topological properties of the underlying network constructed from the input data. Improvements of the low level classification can be achieved by the combination of the two levels of learning. Furthermore, as the class configuration's complexity increases, such as the mixture among different classes, a larger portion of the high level term is required to correctly determine the pattern formation. Particularly and also intuitively, the high level term is not required in the case where all classes are completely separated and each class is well-posed. Finally, we show how the proposed technique is used in a real-world application, where it is able to identify variations of handwritten digits images and, consequently, improve the pattern recognition rate.

5.2 Model Description

In this section, the high level classification technique is presented. Specifically, Section 5.2.1 supplies the general idea of the model and Section 5.2.2 reveals the composition of the proposed hybrid framework.

5.2.1 Glimpsing over the Fundamentals behind the Model

The proposed classifier works in the supervised learning scheme. In this way, consider that it is given a training set denoted here as $\mathcal{X}_{\text{training}} = \{(x_1, y_1), \dots, (x_l, y_l)\}$, where the first component of the i th tuple $x_i = (f_1, \dots, f_d)$ denotes the attributes of the d -dimensional i th training instance. The second component $y_i \in \mathcal{L} = \{L_1, \dots, L_L\}$ characterizes the class label or target associated to that training instance. The goal here is to learn a mapping $x \mapsto y$. Usually, the constructed classifier is checked by using a test set $\mathcal{X}_{\text{test}} = \{x_{l+1}, \dots, x_{l+u}\}$, in which labels are not provided. In this case, each data item is called test instance. For an unbiased learning, the training and test sets must be disjoint, i.e., $\mathcal{X}_{\text{training}} \cap \mathcal{X}_{\text{test}} = \emptyset$.

In the supervised learning scheme, there are two phases of learning: the *training phase* and the *classification phase*. In the training phase, the classifier is induced or trained by using the training instances (labeled data) in $\mathcal{X}_{\text{training}}$. In the classification phase, the labels of the test instances in $\mathcal{X}_{\text{test}}$ are predicted using the induced classifier. Below, these two phases are presented in detail.

Training Phase

In this phase, the data in the training set are mapped into a graph \mathcal{G} using a network formation technique $g : \mathcal{X}_{\text{training}} \mapsto \mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \{1, \dots, V\}$ is the set of vertices and \mathcal{E} is the set of edges. Each vertex in \mathcal{V} represents a training instance in $\mathcal{X}_{\text{training}}$. As it will be described later, the pattern formation of the classes will be

extracted by using the complex topological features of this networked representation. Therefore, the network construction is vital for the prediction produced by the high level classifier.

The edges in \mathcal{E} are created using a combination of the ϵ -radius and k -nearest neighbors (k -NN) graph formation techniques. In the original versions, the ϵ -radius technique creates a link between two vertices if they are within a distance ϵ , while the k -NN sets up a link between vertices i and j if i is one of the k nearest neighbors of j or vice versa. A recent study showed the dramatic influences of these two different graph formation techniques on clustering and classification techniques (Maier and von Luxburg, 2009). Both approaches have their limitations when sparsity or density is a concern. For sparse regions, the k -NN forces a vertex to connect to its k nearest vertices, even if they are far apart. In this scenario, one can say that the neighborhood of this vertex would contain dissimilar points. Equivalently, improper ϵ values could result in disconnected components, subgraphs, or isolated singleton vertices.

In this work, the network is constructed using these two traditional graph formation techniques in a combined form. The neighborhood of a training vertex x_i is given by:

$$\mathcal{N}_{\text{training}}(x_i) = \begin{cases} \epsilon\text{-radius}(x_i, y_{x_i}), & \text{if } |\epsilon\text{-radius}(x_i, y_{x_i})| > k \\ k\text{NN}(x_i, y_{x_i}), & \text{otherwise} \end{cases} \quad (5.1)$$

where y_{x_i} denotes the class label of the training instance x_i , $\epsilon\text{-radius}(x_i, y_{x_i})$ returns the set $\{x_j, j \in \mathcal{V} : d(x_i, x_j) < \epsilon \wedge y_{x_j} = y_{x_i}\}$, and $k\text{NN}(x_i, y_{x_i})$ returns the set containing the k nearest vertices of the same class as x_i . Note that the ϵ -radius technique is used for dense regions ($|\epsilon\text{-radius}(x_i)| > k$), while the k -NN is employed for sparse regions. With this mechanism, it is expected that each class will have a unique and single graph component. Below, we present a simple contextual example showing the proposed network formation technique.

Example 4. Consider the scatter plot in Fig. 5.2, where the task is to determine to which neighbors the central red (dark gray) class vertex will connect. Consider that $k = 2$ and ϵ is the radius illustrated in the figure. As there are $3 > k$ vertices in the ϵ -neighborhood, the area depicted in the figure is considered as a dense region and the modified ϵ -radius technique is employed. In this way, the central red (dark gray) vertex is connected to the other three red (dark gray) vertices reached by this radius.

For the sake of clarity, Fig. 5.3a shows a schematic of how the network looks like for a three-class problem when the training phase is completed. In this case, each class holds a representative component. In the figure, the surrounding circles denote these components: \mathcal{G}_{C_1} , \mathcal{G}_{C_2} , and \mathcal{G}_{C_3} .

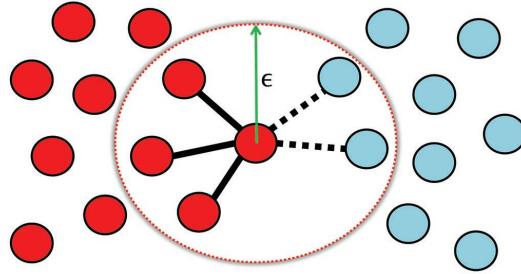


Figure 5.2: Illustration of the graph formation technique consisting in a combination of the k -nearest neighbor and the ϵ -radius techniques. In the depicted network, there are two classes: the red (dark gray) and the blue (light gray) classes. Since $k = 2$, the local region is considered as densely populated. Thus, the ϵ -radius technique is employed. As the centralized vertex belongs to the red (dark gray) class, it is only permitted to be linked to other red (dark gray) class vertices. In this case, no connection will be established between it and the two blue (light gray) vertices.

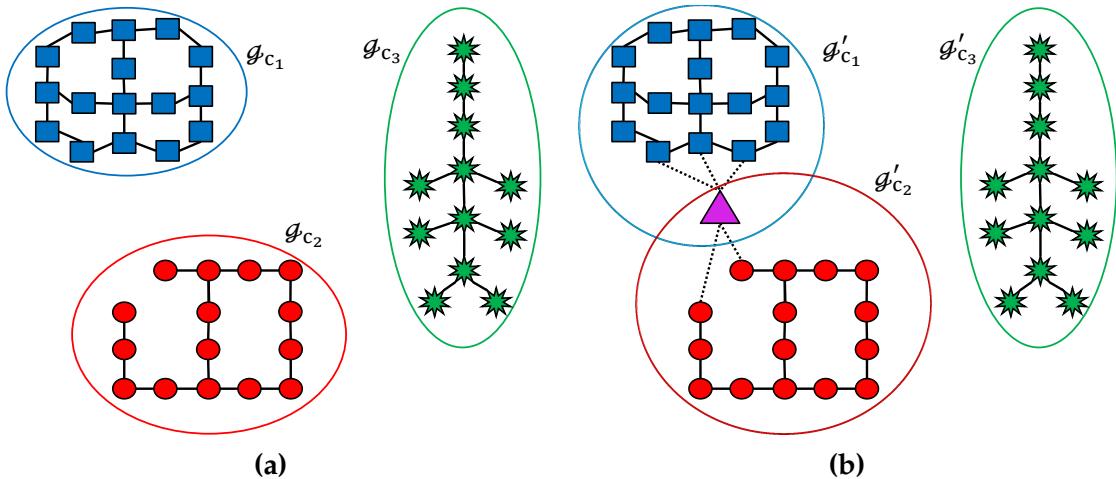


Figure 5.3: Overview of the two phases of the supervised learning. (a) Schematic of the network in the training phase. (b) Schematic of how the classification inference is done.

Classification Phase

In the classification phase, the unlabeled data items in the $\mathcal{X}_{\text{test}}$ are presented to the classifier one by one. In contrast to the training phase, the class labels of the test instances are unknown. In this way, we apply a mixture of the ϵ -radius and the k -NN techniques, but now no information of the classes are used in order to construct the neighborhood. The neighborhood of the test instance x_i is defined as:

$$\mathcal{N}_{\text{classification}}(x_i) = \begin{cases} \epsilon\text{-radius}(x_i), & \text{if } |\epsilon\text{-radius}(x_i)| > k \\ k\text{NN}(x_i), & \text{otherwise} \end{cases} \quad (5.2)$$

meaning that the ϵ -radius connects every vertex within the radius ϵ , disregarding

the class of the neighbor vertices. Likewise the previous case, if the region is sparse enough, i.e., there are less than k vertices in this neighborhood, then the k -NN approach is employed. This prevents test instances from becoming singleton vertices, provided that $k > 0$. Once the data item is inserted, each class analyzes, in isolation, the impact of the insertion of this data item on its respective class component by using a number of complex topological features. In the proposed high level model, each class retains an isolated graph component. Each of these components calculates the changes that occur in its pattern formation with the insertion of this test instance. If slight or no changes occur, then it is said that the test instance is in compliance with that class pattern. As a result, the high level classifier yields a great membership value for that test instance on that class. Conversely, if these changes dramatically modify the class pattern, then the high level classifier produces a small membership value on that class. These changes are quantified via network measures, each of which numerically translating the organization of the component from a local to global fashion. As we will see, the average degree, clustering coefficient, and the assortativity are employed for the high level order of learning.

For the sake of clarity, Fig. 5.3b exhibits a schematic of how the classification process is performed. The test instance (triangle-shaped) is inserted using the traditional ϵ -radius technique. Due to its insertion, the class components become altered: \mathcal{G}'_{C_1} , \mathcal{G}'_{C_2} , and \mathcal{G}'_{C_3} , where each of them is a component surrounded by a circle in Fig. 5.3b. It may occur that some class components do not share any links with this test instance. In the figure, this happens with \mathcal{G}'_{C_3} . In this case, we say that test instance do not comply with the pattern formation of the class component. For the components that share at least a link (\mathcal{G}'_{C_1} and \mathcal{G}'_{C_2}), each of it calculates, in isolation, the impact on its pattern formation by virtue of the insertion of the test instance. For example, when we check the compliance of the test instance with the component \mathcal{G}'_{C_1} , the connections from the test instance to the component \mathcal{G}'_{C_2} are ignored, and vice versa.

Concurrently to the prediction made by the high level classifier, a low level classifier also predicts the membership of the test instance for every class in the problem. The way it predicts depends on the choice of the low level classifier. At the end, the predictions produced by both classifiers are combined via a linear combination to derive the prediction of the high level framework (meta-learning). Once the test instance gets classified, it is either discarded or incorporated to the training set with the corresponding predicted label. In the second case, the classifier must be retrained. Note that, in any of the two situations, each class is still represented by a single graph component.

5.2.2 Deriving the Hybrid Classification Framework

Frequently, the data items present internal structures that form particular patterns. These patterns can range from a class that is characterized by a simple regular network to complex higher dimensional organized lattices disposed throughout the space. These prominent and distinctive qualities that each class may inherently convey are usually ignored by similarity measures that are, ultimately, formed by some sort of physical distance or assumptions underlying the data distribution. In light of that, we introduce a technique that not only can capture the physical distances among the data (as traditional classifiers are able to), but also considers the pattern formation of the data. In a short description, the proposed technique can be comprehended as a high level (semantic) classifier; therefore, it is capable of processing the test set in a more intelligent form using a mixture of local features, provided by the physical distances among the data, and global semantic features, which are artlessly supplied by the class patterns.

With the purpose of representing such mixture, we propose a hybrid classifier F that consists of a convex combination of two terms:

- i. A low level classifier, for instance, a decision tree, SVM, or a k -NN classifier;
- ii. A high level classifier, which is responsible for classifying a test instance according to its pattern formation with the data.

Mathematically, the membership of the test instance $x_i \in \mathcal{X}_{\text{test}}$ with respect to the class $j \in \mathcal{L}$ yielded by the hybrid framework, here written as $F_i^{(j)}$, is given by:

$$F_i^{(j)} = (1 - \rho)L_i^{(j)} + \rho H_i^{(j)}, \quad (5.3)$$

where $L_i^{(j)} \in [0, 1]$ denotes the membership of the test instance x_i towards class j produced by an arbitrary traditional (low level) classifier; $H_i^{(j)} \in [0, 1]$ indicates the same membership information yielded by a high level classifier; and $\rho \in [0, 1]$ is the *compliance term*, which plays the role of counterbalancing the classification decisions supplied by both low and high level classifiers. Whenever $L_i^{(j)} = 1$ and $H_i^{(j)} = 1$, we may deduce that the i th data item carries all the characteristics of class j . On the other hand, whenever $L_i^{(j)} = 0$ and $H_i^{(j)} = 0$, we may infer that the i th data item does not present any similarities nor complies with the pattern formation of class j . Values in-between these two extremes lead to natural uncertainty in the classification process and are found in the majority of times during a classification task. It is worth noting that (5.3) is a convex combination of two scalars, since the sum of the coefficients is unitary. Thus, as the domains of $L_i^{(j)}$ and $H_i^{(j)}$ range from 0 to 1, it is guaranteed that $F_i^{(j)} \in [0, 1]$. Therefore,

(5.3) provides a fuzzy classification method. Moreover, it is valuable to mention that, when $\rho = 0$, (5.3) reduces to a common low level classifier.

A test instance receives the label from the class j that maximizes (5.3). Mathematically, the estimated label of the test instance x_i , \hat{y}_{x_i} , is given by:

$$\hat{y}_{x_i} = \arg \max_{j \in \mathcal{L}} F_i^{(j)}. \quad (5.4)$$

Equation (5.3) supplies a general framework for the hybrid classification process, in the sense that various supervised data classification techniques can be brought into play. The first term of (5.3) is rather straightforward to implement, since it can be any traditional classification technique. The literature provides a myriad of supervised data classification techniques. Some of these include graph-based methods, decision trees, SVM and its variations, neural networks, Bayesian learning, among many others. However, to our knowledge, little has been done in the area of classifiers that take into account the patterns or organizational features inherently hidden into the relationships among the data items. Thus, we now proceed to a detailed analysis of the proposed high level classification term H .

Motivated by the intrinsic ability to describe topological structures among the data items, we propose a network-based (graph-based) technique for the high level classifier H . Specifically, the inference of pattern formation within the data is processed using the generated network. In order to do so, the following structural constraints must be satisfied for any constructed network:

- i. Each class is an isolated subgraph (graph component);
- ii. Each class retains a representative and unique graph component.

With these two network properties in mind, the pattern formation of the data is quantified through a combination of network measures developed in the complex network literature. These measures are chosen in a way to cover relevant high level aspects of the class component, as we will detail later. The number of measures to be plugged into the high level classifier is user-controllable. Here, suppose that m measures, $m > 0$, are selected to comprise the high level classifier H . Mathematically, the membership of the test instance $x_i \in \mathcal{X}_{\text{test}}$ with respect to the class $j \in \mathcal{L}$ yielded by the high level classifier, here written as $H_i^{(j)}$, is given by:

$$H_i^{(j)} = \frac{\sum_{u=1}^m \alpha(u) [1 - f_i^{(j)}(u)]}{\sum_{g \in \mathcal{L}} \sum_{u=1}^m \alpha(u) [1 - f_i^{(g)}(u)]}, \quad (5.5)$$

where $\alpha(u) \in [0, 1], \forall u \in \{1, \dots, m\}$, is a user-controllable coefficient that indicates the influence of each network measure in the classification process and $f_i^{(j)}(u)$ is a function that depends on the u th network measure applied to the i th data item with regard to the class j . This function is responsible for providing an answer whether or not the test instance x_i presents the same patterns or organizational features of the class j . The denominator in (5.5) has been introduced solely for normalization matters. Indeed, in order to (5.5) to be a valid convex combination of network measures, the following constraint must be satisfied:

$$\sum_{u=1}^m \alpha(u) = 1. \quad (5.6)$$

With respect to $f_i^{(j)}(u)$, it possesses a general closed form given by:

$$f_i^{(j)}(u) = \Delta G_i^{(j)}(u) p^{(j)}, \quad (5.7)$$

where $\Delta G_i^{(j)}(u) \in [0, 1]$ is the variation of the u th network measure that occurs on the component representing class j if x_i joins it and $p^{(j)} \in [0, 1]$ is the proportion of data items pertaining to the class j . Remembering that each class has a component representing itself, the strategy to check the pattern compliance of a test instance is to examine whether its insertion causes a great variation of the network measures representing the class component. In other words, if there is a small change in the network measures, the test instance is in compliance with all the other data items that comprise that class component, i.e., it follows the same pattern as the original members of that class. On the other hand, if its insertion is responsible for a significant variation of the component's network measures, then probably the test instance may not belong to that class. This is exactly the behavior that (5.5) together with (5.7) propose, since a small variation of $f(u)$ causes a large membership value output by H ; and vice versa. For didactic purposes, we show this important concept through a simple example, in the following.

Example 5. Suppose a network in which there exists two equally sized classes, namely A and B. For simplicity, let us use a single network measure to quantify the pattern formation ($m = 1$). The goal is to classify a test instance x_i . Hypothetically, say that $\Delta G_i^{(A)}(1) = 0.7$ and $\Delta G_i^{(B)}(1) = 0.3$. In a pattern formation view, x_i has a bigger chance of belonging to class B, since its insertion causes a smaller impact on the pattern formation formed by class B than on the one formed by class A.

Next, we proceed to explain the role of $p^{(j)} \in [0, 1]$ in (5.7). In real-world databases,

unbalanced classes are usually encountered. In general, a database frequently encompasses several classes of different sizes. A great portion of the network measures is very sensitive to the size of the components. In an attempt to soften this problem, we introduce in (5.7) the term $p^{(j)}$, which is the proportion of vertices that class j has. Mathematically, it is given by:

$$p^{(j)} = \frac{1}{V} \sum_{u=1}^V \mathbb{1}_{\{y_u=j\}}, \quad (5.8)$$

where V is the number of vertices and $\mathbb{1}_{\{\cdot\}}$ is the indicator function that yields 1 if the argument is logically true, or 0, otherwise.

In view of the introduction of this mechanism, we expect to obviate the effects of unbalanced classes in the classification process. Again, we present a simple example which evidences the previous concept.

Example 6. Consider a network in which there are two classes: A and B, but now A's size is ten times bigger than B's. From (5.8), $p^{(A)} = 10/11$ and $p^{(B)} = 1/11$. Without the term $p^{(j)}$, it is expected that variations of the network measures in the component A to be considerably smaller than in component B, because of its smaller size. This occurs no matter how the test instance x_i complies with class B. By considering the term $p^{(j)}$, the bigger value of $p^{(A)}$ over $p^{(B)}$ will exactly cancel out the effects of the unbalanced classes in the calculation of the network measures. In this way, this will better reflect the variations that happen in each of the components.

5.3 Some Possible Ways of Composing the High Level Classifier

In our developed works, two different network-based high level classifiers have been proposed. The first one makes use of a mixture of well-known network measures, namely the assortativity, the clustering coefficient, and the average degree measures. The second one uses the dynamical information generated by the tourist walks itself. In the following, we present the two classifiers with the mathematical formalities. Enclosing this section, we show that both approaches are special implementations of the general high level framework.

5.3.1 High Level Classifier using a Mixture of Complex Network Measures

In this section, the first implementation of the high level is introduced, which is composed of three complex network measures, which are: assortativity, clustering coefficient, and average degree. In spite of having chosen these measures, it is worth emphasizing that other network measures can be also plugged into the high level classifier through (5.5). The reason why these three measures have been chosen is as follows: The degree measure figures out strict local scalar information of each vertex in the network; the clustering coefficient of each vertex captures local structures by means of counting triangles formed by the current vertex and any of its two neighbors; the assortativity coefficient considers not only the current vertex and its neighbors, but also the second level of neighbors (neighbor of neighbor), the third level of neighbors, and so on. We can perceive that the three measures characterize the network's topological properties in a local to global fashion. In this way, the combination of these measures is expected to capture the pattern formations of the underlying network in a systematic manner. Below, we present the three measures and how to incorporate them into the hybrid framework.

First Network Measure: Assortativity

The assortativity has been discussed in Section 2.1.5 (first measure). With that in mind, we now derive $\Delta G_i^{(j)}(1)$ using the assortativity measure. Consider that the membership of the test instance x_i with respect to the class j is going to be determined. The actual assortativity measure of the component representing class j (before the insertion of x_i) is given by $r^{(j)}$ (step performed in the training phase). Then, we temporarily insert the data item i into component j using the explained network formation technique (classification phase) and quantify the new component's assortativity measure, here denoted as $r'^{(j)}$. This procedure is performed for all classes $j \in \mathcal{L}$. It may occur that some classes $u \in \mathcal{L}$ will not share any connections with the test instance x_i . Using this approach, $r^{(u)} = r'^{(u)}$, which is undesirable, since this configuration would state that x_i complies perfectly with class u . In order to overcome this problem, a simple post-processing is necessary: for all components $u \in \mathcal{L}$ that do not share at least 1 link with x_i , we deliberately set $r^{(u)} = -1$ and $r'^{(u)} = 1$, i.e., the maximum possible difference. One may interpret this postprocessing as a way to state that x_i do not share any pattern formation with class u , since it is not even connected with it.

In view of this, we are able to calculate $\Delta G_i^{(j)}(1)$ for all $j \in \mathcal{L}$ as follows:

$$\Delta G_i^{(j)}(1) = \frac{|r'^{(j)} - r^{(j)}|}{\sum_{u \in \mathcal{L}} |r'^{(u)} - r^{(u)}|}, \quad (5.9)$$

where the denominator is introduced only for normalization matters. According to (5.9), for components in which the insertion of x_i result in a considerable variation of the assortativity measure, $\Delta G_i^{(j)}(1)$ will be large, and, consequently, by (5.7), $f_i^{(j)}(1)$ is also large. In light of this, the high level classifier H produces a small membership value, as (5.5) reveals. Conversely, for insertions that do not cause a considerable variation of the assortativity, $\Delta G_i^{(j)}(1)$ will be small, resulting in a small $f_i^{(j)}(1)$. As a consequence, the high level classifier H produces a large membership value. In this way, the high level classifier favors test instances that do not impact much the organizational and pattern features of a class.

Second Network Measure: Clustering Coefficient

The clustering coefficient has been discussed in Section 2.1.5 (second measure). With that in mind, we motivate the use of the clustering coefficient by eliciting the following facts: components with large clustering coefficient are found to have a modular structure with a high density of local connections, while components with small average clustering values tend to have many long-range connections, with the absence of local structures.

The derivation of $\Delta G_i^{(j)}(2)$ is rather analogous to the previous case, except for a simple detail: In this case, for all components $u \in \mathcal{L}$ that do not share at least 1 link with the test instance x_i , we intentionally fix $CC^{(u)} = 0$ and $CC'^{(u)} = 1$, i.e., the maximum possible difference, since $CC^{(u)}$ ranges from $[0, 1]$. In this way, we are able to define $\Delta G_i^{(j)}(2)$ as:

$$\Delta G_i^{(j)}(2) = \frac{|CC'^{(j)} - CC^{(j)}|}{\sum_{u \in \mathcal{L}} |CC'^{(u)} - CC^{(u)}|}. \quad (5.10)$$

Third Network Measure: Average Degree

The average degree measure has been discussed in Section 2.1.5 (third measure). The average degree measure is relatively a simple measure, which statistically quantifies the average degree of the vertices of a component. This measure by itself is weak in terms of finding patterns in the network, since the mean value may not exactly quantify the degrees of the majority of vertices in a component. However, if it is jointly used with other measures, such as assortativity and clustering coefficient, its recogni-

tion power significantly increases.

The derivation of $\Delta G_i^{(j)}(3)$ is similar to the previous case, except for a simple particularity: For all components $u \in \mathcal{L}$ that do not share at least 1 link with test instance x_i , we purposefully assign $\langle k'^{(u)} \rangle = \max \left(\langle k^{(u)} \rangle - \min_i \left(k_i^{(u)} \right), \max_i \left(k_i^{(u)} \right) - \langle k^{(u)} \rangle \right)$, i.e., the maximum possible difference from the mean of the component. In this way, we are able to define $\Delta G_i^{(j)}(3)$ as:

$$\Delta G_i^{(j)}(3) = \frac{|\langle k'^{(u)} \rangle - \langle k^{(u)} \rangle|}{\sum_{u \in \mathcal{L}} |\langle k'^{(u)} \rangle - \langle k^{(u)} \rangle|}. \quad (5.11)$$

Again, $\langle k'^{(u)} \rangle$ and $\langle k^{(u)} \rangle$ represent the average degree of the component u before and after the test instance x_i is inserted, respectively.

5.3.2 High Level Classifier using Tourist Walks

In this section, we introduce a second approach for implementing the high level classifier. Specifically, we will utilize the dynamics generated by a tourist walk released into a network. The tourist walks have already been described in Section 2.1.5. For the sake of clarity, we retrieve, in a synthetic manner, the principal components that are going to be utilized in this section.

A tourist walk can be conceptualized as a walker (tourist) aiming at visiting sites (data items) in a d -dimensional map, representing the data set. At each discrete time step, the tourist follows a simple deterministic rule: it visits the nearest site which has not been visited in the previous μ steps. In other words, the walker performs partially self-avoiding deterministic walks over the data set, where the self-avoiding factor is limited to the memory window $\mu - 1$. This quantity can be understood as a repulsive force emanating from the sites in this memory window, which prevents the walker from visiting them in this interval (refractory time). Each tourist walk can be decomposed in two terms: (i) the initial *transient part* of length t and (ii) a *cycle* (attractor) with period c .

Before going any further, one must know that, in the majority of the works related to these walks (Kinouchi *et al.*, 2002; Lima *et al.*, 2001; Stanley and Buldyrev, 2001), the tourist may visit any other site other than the ones contained in its memory window. As μ increases, there is a significant chance that the walker will begin performing large jumps in the data set, since the neighborhood is most likely to be already visited in its entirety within the time frame μ . As we will show in this thesis, in the context of classification, this is an undesirable characteristic that can be simply avoided by using a graph representation of the input data. In this way, the walker is only permitted

to visit vertices, represented now by the sites, that are in its connected neighborhood (link). With this modified mechanism, it is most probable that, for large values of μ , depending on the network configuration, the walker will get trapped within a vertex, not being able to further visit other vertices of the neighborhood. In this scenario, we say that the walk only had a transient part and the cycle period is null ($c = 0$). Therefore, the tourist walks applied to a networked environment is a relatively new approach taken here. Additionally, its utilization for discovering patterns in a network is a totally novel scheme in the literature.

Having in mind the basic concepts revolving around tourist walks, we proceed to explain the high level classifier based on it. In mathematical terms, its decision output is given by:

$$H_i^{(j)} = \frac{\sum_{\mu=0}^{\mu_c} [\alpha_t(1 - T_i^{(j)}(\mu)) + \alpha_c(1 - C_i^{(j)}(\mu))] }{\sum_{g \in \mathcal{L}} \sum_{\mu=0}^{\mu_c} [\alpha_t(1 - T_i^{(g)}(\mu)) + \alpha_c(1 - C_i^{(g)}(\mu))] } \quad (5.12)$$

where μ_c is a critical value that indicates the maximum memory length of the tourist walks, $\alpha_t, \alpha_c \in [0, 1]$ are user-controllable coefficients that indicate the influence of each network measure in the process of classification, $T_i^{(j)}(\mu)$ and $C_i^{(j)}(\mu)$ are functions that depend on the transient and cycle lengths, respectively, of the tourist walk applied to the i th data item with regard to the class j . These functions are responsible for providing an estimative whether or not the data item i under analysis possesses the same patterns of the component j . The denominator in (5.12) has been introduced solely for normalization matters. Indeed, in order to (5.12) to be a valid convex combination of network measures, α_t and α_c must be chosen such as to satisfy $\alpha_t + \alpha_c = 1$.

Regarding $T_i^{(j)}(\mu)$ and $C_i^{(j)}(\mu)$, they are given by the following expressions:

$$\begin{aligned} T_i^{(j)}(\mu) &= \Delta t_i^{(j)}(\mu) p^{(j)} \\ C_i^{(j)}(\mu) &= \Delta c_i^{(j)}(\mu) p^{(j)} \end{aligned} \quad (5.13)$$

where $\Delta t_i^{(j)}(\mu), \Delta c_i^{(j)}(\mu) \in [0, 1]$ are the variations of the transient and cycle lengths that occur on the component representing class j if i joins it and $p^{(j)} \in [0, 1]$ is the proportion of data items pertaining to class j .

Next, we explain how to compute $\Delta t_i^{(j)}(\mu)$ and $\Delta c_i^{(j)}(\mu)$ that appear in (5.13). Firstly, we need to numerically quantify the transient and cycle lengths of a component. Since the tourist walks are strongly dependent on the starting vertices, for a fixed μ , we perform tourist walks initiating from each one of the vertices that are members of a class component. The transient and cycle lengths of the j th component, $\langle t_i^{(j)} \rangle$ and $\langle c_i^{(j)} \rangle$, are

simply given by the average transient and cycle lengths of all its vertices, respectively. In order to estimate the variation of the component's network measures, consider that $x_i \in \mathcal{X}_{\text{test}}$ is a test instance. In relation to an arbitrary class j , we virtually insert x_i into component j using the network formation technique that we have seen, and re-calculate the new average transient and cycle lengths of this component. We denote these new values as $\langle t'_i^{(j)} \rangle$ and $\langle c'_i^{(j)} \rangle$, respectively. This procedure is performed for all classes $j \in \mathcal{L}$. It may occur that some classes $u \in \mathcal{L}$ will not share any connections with the test instance x_i . Using this approach, $\langle t_i^{(k)} \rangle = \langle t'_i^{(k)} \rangle$ and $\langle c_i^{(k)} \rangle = \langle c'_i^{(k)} \rangle$, which is undesirable, since this configuration would state that x_i complies perfectly with class u . In order to overcome this problem, a simple postprocessing is necessary: For all components $u \in \mathcal{L}$ that do not share at least 1 link with x_i , we deliberately set $\langle t'_i^{(j)} \rangle$ and $\langle c'_i^{(j)} \rangle$ to a high value. This high value must be greater than the largest variation that occurs in a component which shares a link with the data item under analysis. One may interpret this postprocessing as a way to state that x_i does not share any pattern formation with class u , since it is not even connected to it.

With all this information at hand, we are able to calculate $\Delta t_i^{(j)}(\mu)$ and $\Delta c_i^{(j)}(\mu)$, $\forall j \in \mathcal{L}$, as follows:

$$\begin{aligned}\Delta t_i^{(j)}(\mu) &= \frac{|\langle t'_i^{(j)} \rangle - \langle t_i^{(j)} \rangle|}{\sum_{u \in \mathcal{L}} |\langle t'_i^{(u)} \rangle - \langle t_i^{(u)} \rangle|} \\ \Delta c_i^{(j)}(\mu) &= \frac{|\langle c'_i^{(j)} \rangle - \langle c_i^{(j)} \rangle|}{\sum_{u \in \mathcal{L}} |\langle c'_i^{(u)} \rangle - \langle c_i^{(u)} \rangle|}\end{aligned}\tag{5.14}$$

where the denominator is introduced only for normalization matters. According to (5.14), for insertions that result in a considerable variation of the component's transient and cycle lengths, $\Delta t_i^{(j)}(\mu)$ and $\Delta c_i^{(j)}(\mu)$ will be high. In view of (5.13), $T_i^{(j)}(\mu)$ and $C_i^{(j)}(\mu)$ are expected to be also high, yielding a low membership value predicted by the high level classifier $H_i^{(j)}$, as (5.12) reveals. On the other hand, for insertions that do not significantly interfere in the pattern formation of the data, $\Delta t_i^{(j)}(\mu)$ and $\Delta c_i^{(j)}(\mu)$ will be low, and, as a result, $T_i^{(j)}(\mu)$ and $C_i^{(j)}(\mu)$ are expected to be also low, producing a high membership value for the high level classifier $H_i^{(j)}$, as (5.12) exposes.

The network-based high level classifier quantifies the variations of the transient and cycle lengths of tourist walks with limited memory μ that occur in the class components when a test instance artificially joins each of them in isolation. According to (5.12), this procedure is performed for several values of the memory length μ , ranging from 0 (memoryless) to a critical value μ_c . This is done in order to capture complex patterns of each of the representative class components in a local to global fashion. When μ is small, the walks tend to possess a small transient and cycle parts, so that

the walker does not wander far away from the starting vertex. In this way, the walking mechanism is responsible for capturing the local structures of the class component. On the other hand, when μ increases, the walker is compelled to venture deep into the component, possibly very far away from its starting vertex. In this case, the walking process is responsible for capturing the global features of the component. In summary, the fundamental idea of the high level classifier is to make use of a mixture of local and global features of the class components by means of a combination of tourist walks with different values of μ .

One may wonder how the high level classifier based on tourist walks given by (5.12) is plugged into the hybrid framework, whose general equations are given by (5.5) and (5.7). The following proposition links both approaches.

* * *

Proposition 3. *The high level classifier based on tourist walks, whose decision equations are:*

$$H_i^{(j)} = \frac{\sum_{\mu=0}^{\mu_c} [\alpha_t(1 - T_i^{(j)}(\mu)) + \alpha_c(1 - C_i^{(j)}(\mu))] }{\sum_{g \in \mathcal{L}} \sum_{\mu=0}^{\mu_c} [\alpha_t(1 - T_i^{(g)}(\mu)) + \alpha_c(1 - C_i^{(g)}(\mu))]}, \quad (5.15)$$

$$\begin{aligned} T_i^{(j)}(\mu) &= \Delta t_i^{(j)}(\mu) p^{(j)} \\ C_i^{(j)}(\mu) &= \Delta c_i^{(j)}(\mu) p^{(j)}, \end{aligned} \quad (5.16)$$

is a particular form of the generic high level framework given in:

$$H_i^{(j)} = \frac{\sum_{u=1}^m \alpha(u) [1 - f_i^{(j)}(u)]}{\sum_{g \in \mathcal{L}} \sum_{u=1}^m \alpha(u) [1 - f_i^{(g)}(u)]}, \quad (5.17)$$

$$f_i^{(j)}(u) = \Delta G_i^{(j)}(u) p^{(j)}. \quad (5.18)$$

Proof. First, one can see that (5.15) can be written as:

$$H_i^{(j)} = \frac{\sum_{\mu=1}^{2\mu_c+2} v(\mu) (1 - V_i^{(j)}(\mu))}{\sum_{g \in \mathcal{L}} \sum_{\mu=1}^{2\mu_c+2} v(\mu) (1 - V_i^{(g)}(\mu))}, \quad (5.19)$$

where:

$$v(\mu) = \begin{cases} \alpha_t & \text{if } \mu \leq \mu_c + 1 \\ \alpha_c & \text{if } \mu > \mu_c + 1 \end{cases}, \quad (5.20)$$

and

$$V_i^{(j)}(\mu) = \begin{cases} T_i^{(j)}(\mu) & \text{if } \mu \leq \mu_c + 1 \\ C_i^{(j)}(\mu) & \text{if } \mu > \mu_c + 1 \end{cases}, \quad (5.21)$$

are piecewise functions defined over the domain $\{1, \dots, 2\mu_c + 2\}$. In view of (5.6), one must have that:

$$\sum_{\mu=1}^{2\mu_c+2} v(\mu) = 1 \Rightarrow (\mu_c + 1)(\alpha_t + \alpha_c) = 1. \quad (5.22)$$

Moreover, by equivalence, comparing the upper limits of the summations of (5.17) and (5.19), one has that:

$$m = 2\mu_c + 2. \quad (5.23)$$

Using (5.6) on (5.22), one has:

$$(\mu_c + 1)(\alpha_t + \alpha_c) = \sum_{\mu=1}^{2\mu_c+2} v(\mu) = 1 = \sum_{u=1}^m \alpha(u) = \sum_{u=1}^{2\mu_c+2} \alpha(u). \quad (5.24)$$

Therefore, by polynomial equivalence, one can take:

$$\alpha(u) = v(u) = \begin{cases} \alpha_t & \text{if } u \leq \mu_c + 1 \\ \alpha_c & \text{if } u > \mu_c + 1 \end{cases}. \quad (5.25)$$

Using the same reasoning, we have that:

$$f_i^{(j)}(u) = V_i^{(j)}(u) = \begin{cases} T_i^{(j)}(u) & \text{if } u \leq \mu_c + 1 \\ C_i^{(j)}(u) & \text{if } u > \mu_c + 1 \end{cases}. \quad (5.26)$$

Finally, by looking at (5.16) and (5.18), we can infer that the tourist walk based classifier can be coupled into the hybrid framework by taking $2\mu_c + 2$ network measures, such that the first $\mu_c + 1$ are the transient lengths with increasing memory lengths, each of which weighted by α_t , and the remaining $\mu_c + 1$ are cycle lengths with increasing memory lengths, each of which weighted by α_c . In view of this, the high level classifier based on tourist walks is, in fact, a particular implementation of the generic high level classifier. \square

* * *

5.4 Computer Simulations using the High Level Classifier as a Mixture of Complex Network Measures

In this section, computer simulations are conducted in order to assess the effectiveness of the proposed high level classification model. Here, we utilize the combination of the assortativity, clustering coefficient, and the average degree measures to compose the high level term. Specifically, Section 5.4.1 provides simulations on artificial data sets and a detailed analysis of how the model behaves for different compliance terms, whereas Section 5.4.2 supplies the performance of the proposed model on real-world data sets. In all simulations, the similarity measure used in the network construction is given by the reciprocal of the Euclidean distance.

5.4.1 Illustrative Examples

For the sake of clarity, in this section, the influences of the three complex network measures in the decision produced by the high level classifier are set to be the same, i.e., $\alpha(1) = \alpha(2) = \alpha(3) = 1/3$, according to (5.5). Note that the constraint imposed by (5.6) is satisfied. Mainly, we will focus on particular situations where low level classifiers would have trouble in correctly classifying the data items in the test set. Therefore, this section serves as a tool for better motivating the usage of the proposed model.

In accordance with (5.1), the network in the training phase is constructed using a combination of modified versions of the ϵ -radius and the k -NN graph formation techniques. Here, we set $k = 1$ and $\epsilon = 0.05$. The same parameters are used in the classification phase (see (5.2)). With respect to the low level classifier, a fuzzy SVM classifier is employed (Lin and Wang, 2002), whose optimization method criterion is defined as the Karush-Kuhn-Tucker violation fixed at 10^{-3} (the same condition used in (Hsu and Lin, 2002)). In our first example, 4 independent runs are performed, each of which using a different kernel with an optimized parameter configuration. The used kernels are: Linear, Radial Basis Function (RBF), Sigmoid, and Polynomial (Vapnik,

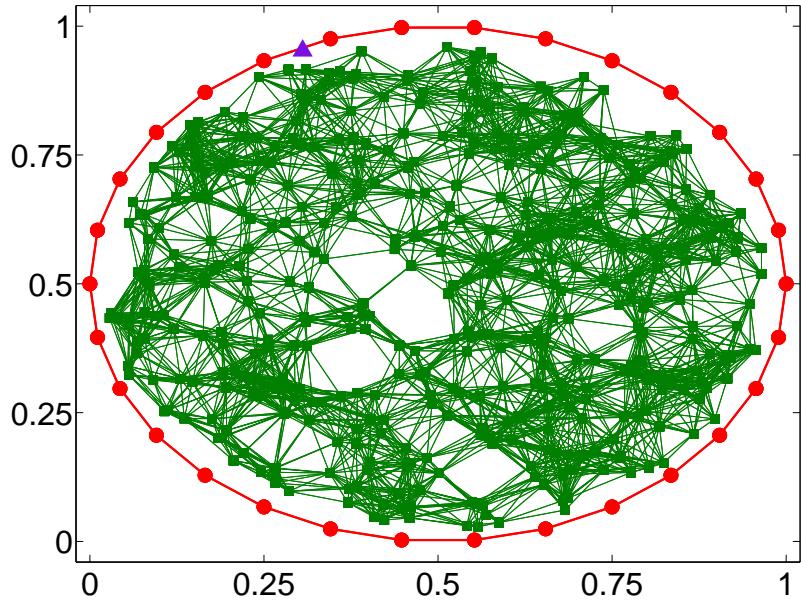


Figure 5.4: A high level data classification using a synthetic data set. The big triangle-shaped data item is to be classified. There are 2 classes in the problem: A sparse hollow rounded class (“circle” or red class) and a condensed opaque rounded class (“square” or green class).

1998). Say that v and u are two data items, then the aforementioned kernels have the following formula:

- i. Linear: $u \cdot v$;
- ii. RBF: $\exp(-\gamma \| u - v \|^2)$;
- iii. Sigmoid: $\tanh(\gamma u \cdot v + c)$;
- iv. Polynomial: $(c + u \cdot v)^d$.

Additionally, the cost of the regularizer parameter C must be fixed. The parameters are optimized over the sets: $\gamma = \{2^4, 2^3, \dots, 2^{-10}\}$, $c = \{10, 9.5, \dots, -10\}$, $d = \{10, 9, \dots, 1\}$, and $C = \{2^{12}, 2^{11}, \dots, 2^{-2}\}$. The optimization process consists in taking every combination of parameters in the sets previously shown. The best model is the one that yields the smallest error. The error is estimated using a stratified ten-fold cross-validation.

As an introductory example, consider the toy data set depicted in Fig. 5.4, where there are two classes, namely the red or “circle” (23 vertices) and the green or “square” (276 vertices) classes. Consequently, the proportions of vertices of each class are $p^{(\text{red})} = 7.69\%$ and $p^{(\text{green})} = 92.31\%$. Clearly, the red class carries a perceivable pattern, which, geometrically speaking, is a “circle” disposed throughout the space. On the other hand, the green class shows no clear pattern (well-defined pattern). The task

Table 5.1: Results obtained for the classification of the triangle-shaped data item in Fig. 5.4.

Kernel	Optimized Parameters (SVM)				Estimated Membership Values				Final Prediction						
	Low Level Classif.	High Level Classif.	$\rho = 0$	$\rho = 0.5$	$\rho = 0.8$	($L_i^{(\cdot)}$)	($H_i^{(\cdot)}$)	Red	Green	Red	Green	Red	Green		
Linear	2^4	n/a	n/a	n/a	n/a	0.04	0.96	0.58	0.42	0.04	0.96	0.31	0.69	0.53	0.47
RBF	2^4	2^{-2}	n/a	n/a	n/a	0.31	0.69	0.86	0.14	0.31	0.69	0.59	0.42	0.81	0.20
Sigmoid	2^1	2^{-1}	-3.00	n/a	n/a	0.27	0.73	0.77	0.23	0.27	0.73	0.52	0.48	0.72	0.28
Polynomial	2^2	2^{-1}	-1.50	7	0.22	0.78	0.73	0.27	0.22	0.78	0.48	0.53	0.68	0.32	

is to classify the big “triangle” data item in either of these two classes. In this simulation, we show that the obtained results are not satisfactory if we only apply the fuzzy SVM with several well-known kernels. Even though the kernel function is responsible for spawning the data items into a higher space through a nonlinear transformation, the data items in this and the next simulations may only be distinguished by using semantic knowledge, i.e., the pattern formation of the classes. It is worth stressing that special kernels could be designed in order to solve specific data distributions, but they would be totally peculiar for the problems at hand. So, their usability is constrained to a very few real situations. On the other side, as it will be stressed throughout this work, the proposed technique captures the pattern formation of the classes in a general manner. Table 5.1 shows the classification results of the triangle-shaped data item in Fig. 5.4 for all the four employed kernels. By inspecting the estimated membership value section of the table, one can confirm that, while the low level classifier always decides favorable to the green class, the high level classifier, due to the strong pattern exhibited by the red class, decides favorable to it. The final prediction is given by a weighted combination of the decisions of the low and high level classifiers for three different compliance terms: $\rho = 0$ (pure SVM), $\rho = 0.5$ (equal weight for the decision of the SVM and the high level classifier), and $\rho = 0.8$ (a high weight for the high level classifier and a low weight for the SVM). We can clearly see that a pure SVM decision ($\rho = 0$), no matter which kernel is used, is not able to correctly classify the data item shown in Fig. 5.4. However, if the final decision of the classifier is based on a mixture of the SVM and the high level classifier, one can verify that correct results can be obtained. As ρ increases, the proposed classifier tends to put the triangle-shaped data item in the red or “circle” class, by virtue of the strong compliance that it forms with the data item.

Now, we move on to an interesting analysis of the decision boundaries delineated by the proposed technique on structured data. This example serves as the gist of how the proposed classifier draws its decisions. Consider the toy data set illustrated in Fig. 5.5, in which there are 2 classes: a red or “circle” (35 vertices) and a green or “square” (120 vertices). The network construction in the training and classification phases remains the same as the previous example. The fuzzy SVM with RBF kernel ($C = 100$ and $\gamma = 2^{-2}$) is adopted for the low level classifier. By inspection of the figure, the red or “circle” class also shows a strong clear pattern: a grid or lattice, whereas the green or “square” class does not indicate any clear patterns. The goal is to classify the triangle-shaped data items (test set) one by one only using the information of the training set (fully supervised learning). Figures 5.6a, 5.6b, and 5.6c exhibit the decision boundaries of the two classes when $\rho = 0$, $\rho = 0.5$, and $\rho = 0.8$, respectively. When $\rho = 0$, only the prediction of the fuzzy SVM is used by the proposed technique. In this case, one can see that the five data items are not correctly classified. Notice that

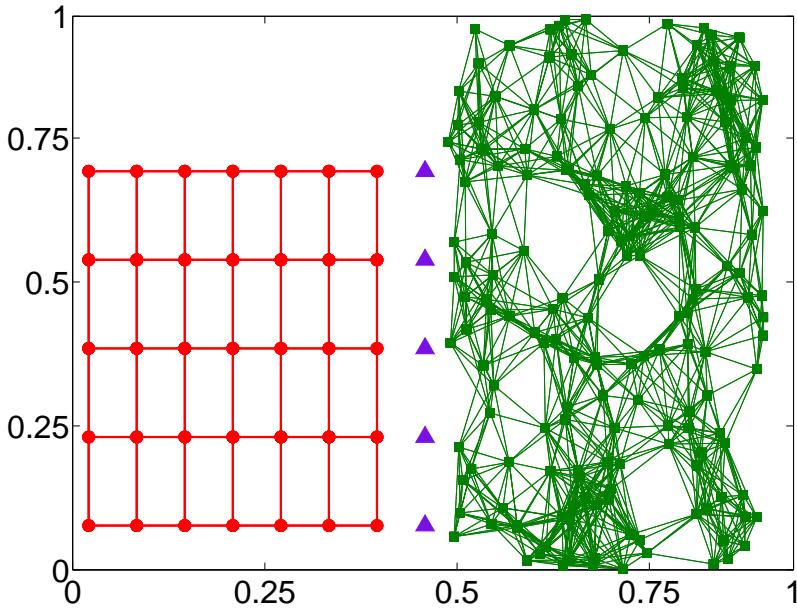


Figure 5.5: Another example of high level classification in a synthetic data set. The toy data set comprises two classes: a well-structured two-dimensional lattice class (“circle” or red class) and a condensed opaque squared class (“square” or green class)

the decision boundaries are established near the red or “circle” class by virtue of the large amount of green or “square” items in the vicinity. Now, when $\rho = 0.5$, the SVM and the high level classifier predictions are used in the same intensity. In this case, the decision borders are dragged toward the green or “square” class, because of the strong pattern that the red or “square” class exhibits. When $\rho = 0.8$, the decision derived by the high level classifier is so strong that is capable of pushing the decision boundaries inside the high density area of the green or “square” class. This happens on account of the strong pattern that the red or “square” class shows. In the two former cases, the proposed technique can successfully classify the triangle-shaped data items. In summary, the concept of classification is altered depending on the value of the compliance term. A small compliance term makes the decisions be derived from traditional assumptions about the data items, such as proximity or cluster assumptions. When a large compliance term is used, the concept that the proposed classifier tries to uncover is the patterns that the classes show. As the pattern of a class gets more structured and easier to be evidenced, the wider will be decision boundaries delineated by that class.

Let us now consider a multiclass problem. Consider the toy data set shown in Fig. 5.7a, where, visually, we see four classes: the red or “circle,” the purple or “star,” the blue or “square,” and the green or “diamond” classes. The network construction in the training and classification phases remains the same as the previous example. Again, the fuzzy SVM with RBF kernel ($C = 2$ and $\gamma = 2^4$) is used for the low level classifier. Figures 5.7b and 5.7c display the corresponding results when $\rho = 0$ and $\rho = 0.2$, re-

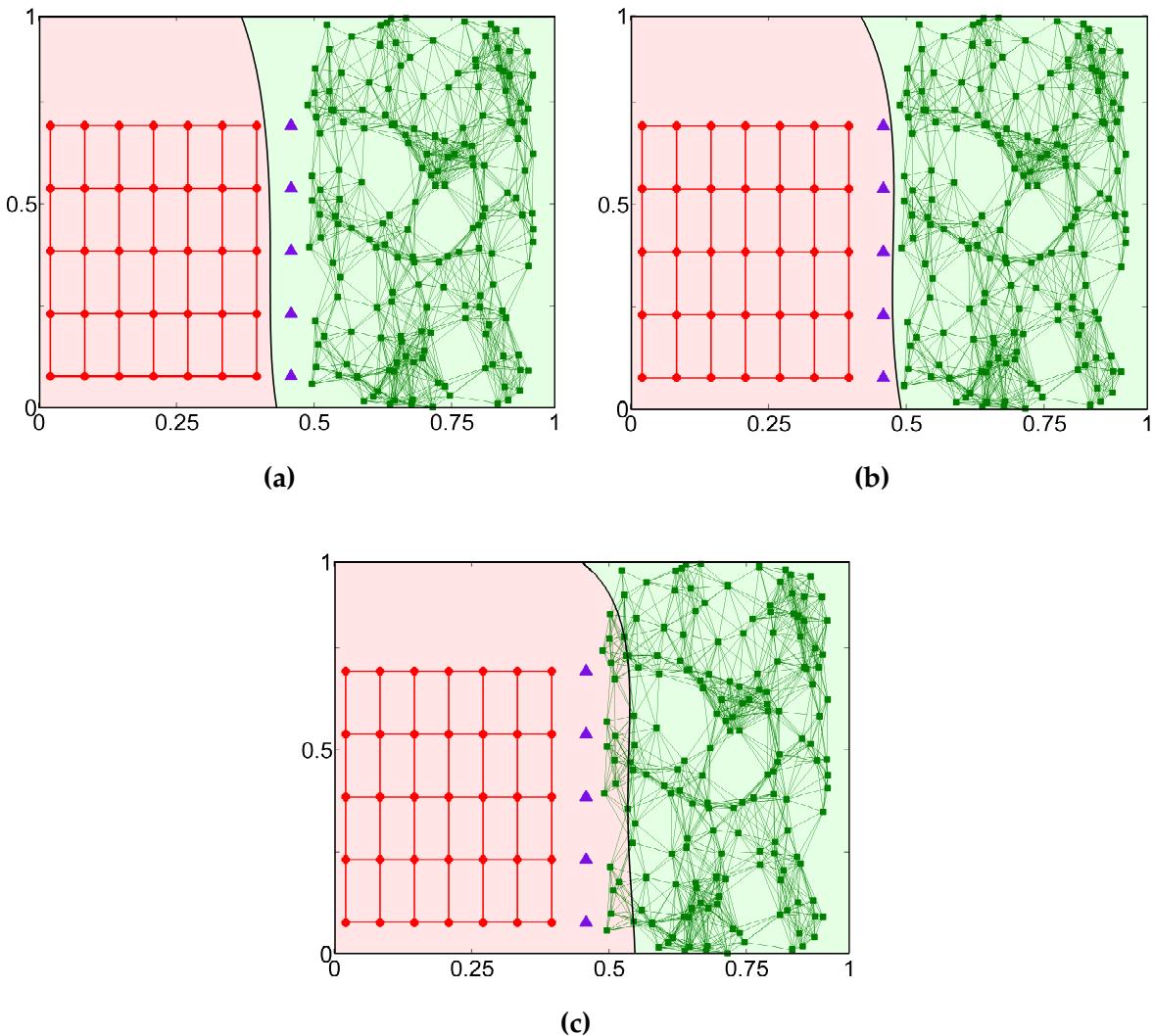


Figure 5.6: Behavior of the decision boundaries as ρ varies in the toy data of Fig. 5.5. Decision boundaries when (a) $\rho = 0$; (b) $\rho = 0.5$; and (c) $\rho = 0.8$.

spectively. From Figs. 5.7b, we see that the decision boundaries of the blue or “square” class invade the red or “circle” class. As a consequence, almost all the red or “circle” instances will be wrongly classified. A similar situation occurs with the other two classes, where the decision boundaries of the green or “diamond” class partially invade the purple or “star” class. On the other hand, when $\rho = 0.2$ (a small portion of high level classification is taken into consideration), we see that the decision boundaries are well-located, which means that most of the class instances can be correctly classified.

Next, we will study problematic classification situations, where a low level classifier can be deceived if it only considers the density in the surroundings to perform predictions. In this context, we will encounter the minimum required compliance term, ρ_{\min} , for which the proposed technique can satisfactorily avoid such traps. Consider

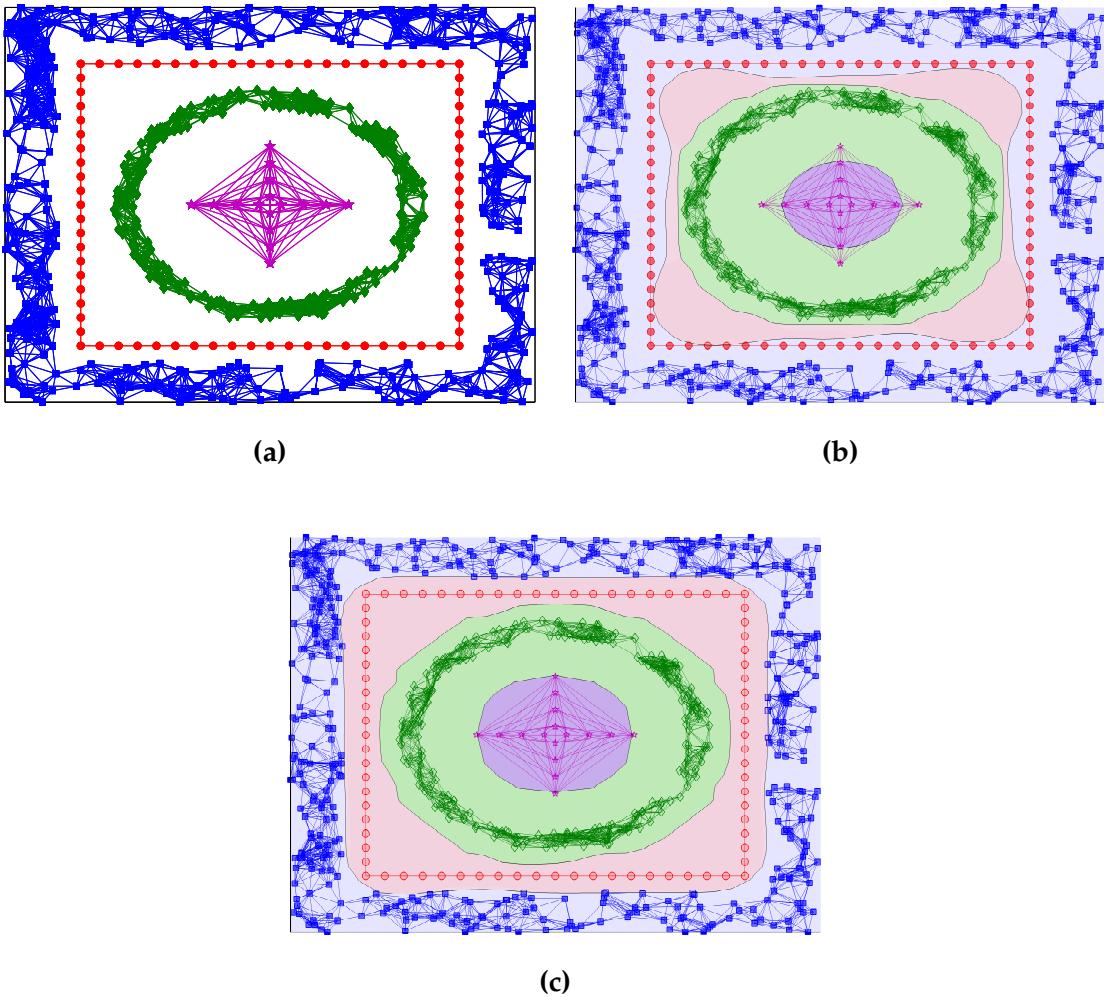


Figure 5.7: Behavior of the decision boundaries as ρ varies for a multiclass problem. (a) Scatter plot of the toy data set. Decision boundaries when (b) $\rho = 0$; and (c) $\rho = 0.2$.

the classification problems arranged in Figs. 5.8a and 5.8b, where the test sets are: a perfect straight line and a noisy line, respectively. In each figure, there are two classes in the problem: the red or "circle" class (7 vertices) representing a strong pattern of a segment of line and a condensed rectangular class outlined by the green or "square" class (100 vertices). The network formation parameters in the training and classification phases are set to $k = 1$ and $\epsilon = 0.07$ (this radius covers, for any vertex in the straight line, 2 adjacent vertices, except for the vertices at each end). The fuzzy SVM with RBF kernel ($C = 2^8$ and $\gamma = 2^4$) is adopted as the low level classifier. The task is to classify the 8 test data items depicted by the big "triangles." In this case, after a data item in the test set is classified, it is incorporated to the training set with the predicted label and the low and high level classifiers are retrained. The orientation is set from left to right. The aligned graphics embedded in Figs. 5.8a and 5.8b show the minimum compliance terms required for the test instances to be classified as members of the red class. For the first one, $\rho = 0$ suffices, i.e., the fuzzy SVM technique is enough

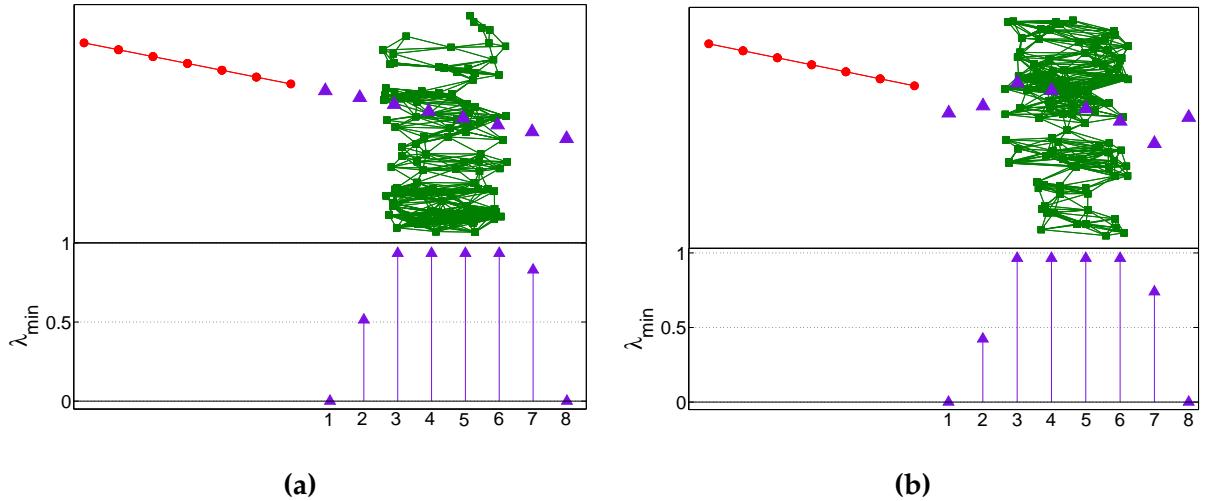


Figure 5.8: Analysis of the minimum value of the compliance term, ρ_{\min} , that classifies the triangle-shaped item as members of the red or “circle” class. If a smaller value of ρ is chosen, then the test data items are classified as members of the green or “square” class. The orientation was set from left to right. Whenever a data item is classified, it is incorporated to the corresponding most similar class. Traditional techniques would definitely fail to correctly classify the straight line that diametrically crosses the densely connected component pertaining to the green or “square” class. (a) A test set encompassing a perfect straight line. (b) A test set encompassing a noisy “straight” line.

to produce the right prediction. However, from the second to the seventh, nonzero ρ_{\min} are required. Specifically, as the perfect or noisy line diametrically crosses the condensed region pertaining to the green or “square” class, $\rho_{\min} \rightarrow 1$. In this situation, if one expects to find pattern formation within the data, one cannot establish its decision based on the low level classifier, because it would erroneously decide favorable to the green or “square” class due to the high deceiving density of samples in the region. From this scenario, one can see that the fuzzy SVM or its variations classify the test instances that are deep within the densely rectangular green or “square” class as members of the green or “square” class. This holds even if nonlinear transformations to map the data into higher space are employed. This happens because there is no general way to separate the test instances and the mass of green or “square” neighbors by using class topologies. Here, the pattern formation can help in solving this hard problem. Moreover, it is worth mentioning that even manifold learning techniques, such as Isomap, Laplacian eigenmaps, Locally-Linear Embedding techniques, are not able to correctly map the same items, since they generally attempt to find a nonlinear transformation which fits the data into a lower dimensional space (manifold assumption), always preserving the local relationships. In other words, data items, which are close in the original space, tend to stay even closer in the resulting reduced dimension. In the case of Figs. 5.8a and 5.8b, this would make the problem even harder to be solved.

In brief, a high level order of learning is required to successfully classify these types of data items. The proposed high level classifier is an attempt to fill this gap. Figure 5.8b presents the detection of a “noisy” straight line, which illustrates the robustness of the proposed technique.

In order to give a clear idea of how the complex network measures vary, Table 5.2 reports the variations of the three network measures employed by the high level classifier and the corresponding prediction for $\rho = 0$ and $\rho = 0.95$, when the first 5 left-most triangle-shaped data items in Fig. 5.8a are classified. A careful look at Table 5.2 corroborates the importance that the proportion of vertices plays in the inference process, as it is responsible for counterbalancing the nominal variations of each network measure against the size of the component. In this case, the small variations that occur due to the insertion of a new data item in the green or “square” class are amplified by the high value of $p^{(\text{green})}$. Conversely, the sizable variations that take place in the red or “circle” class by virtue of the addition of a new data item are softened by the low value of $p^{(\text{red})}$.

Table 5.2: Snapshot of the measures captured in the contextual classification task of the big “triangle” data items in Fig. 5.8a. The items are enumerated from left to right (orientation). The left-most “triangle” data item is the item 1 and so forth, up to the fifth item (number 5). Bolded values in the two right-most columns emphasize the final classification of the algorithm for $\rho = 0$ and $\rho = 0.95$.

Item	Class	Calculated Network Measures								Final Decision		
		Assortativity			Clustering Coeff.			Mean Degree		$\rho = 0$		
		$r^{(.)}$	$r'^{(.)}$	$\Delta G_i^{(.)}(1)$	$CC^{(.)}$	$CC'^{(.)}$	$\Delta G_i^{(.)}(2)$	$\langle k^{(.)} \rangle$	$\langle k'^{(.)} \rangle$	$\Delta G_i^{(.)}(3)$	$\rho = 0.95$	
1	Red	0.9519	0.9597	0.0000 ⁷	0.0000	0.0000	0.0000 ⁷	1.7143	1.7500	0.0000 ⁷	1.0000	1.0000
	Green	0.5626	0.5626	1.0000 ⁷	0.6745	0.6745	1.0000 ⁷	9.5489	9.5489	1.0000 ⁷	0.0000	0.0000
2	Red	0.9597	0.9653	0.4095	0.0000	0.0000	0.0000	1.7500	1.7778	0.3377	0.1973	0.7518
	Green	0.5626	0.5707	0.5905	0.6745	0.6651	1.0000	9.5489	9.4944	0.6623	0.8027	0.2482
3	Red	0.9653	0.9695	0.3936	0.0000	0.0000	0.0000	1.7778	1.8000	0.3735	0.0036	0.7406
	Green	0.5626	0.5691	0.6064	0.6745	0.6696	1.0000	9.5489	9.5116	0.6265	0.9964	0.2594
4	Red	0.9695	0.9728	0.3070	0.0000	0.0000	0.0000	1.8000	1.8182	0.3897	0.0289	0.7544
	Green	0.5626	0.5701	0.6930	0.6745	0.6705	1.0000	9.5489	9.5204	0.6103	0.9711	0.2456
5	Red	0.9728	0.9755	0.4794	0.0000	0.0000	0.0000	1.8182	1.8333	0.8718	0.0004	0.6480
	Green	0.5626	0.5655	0.5206	0.6745	0.6719	1.0000	9.5489	9.5511	0.1282	0.9996	0.3520

As a last analysis in this experimental section, the behavior of the accuracy rate achieved by the proposed technique on Gaussian-distributed classes for varying values of ρ will be studied. Firstly, a series of multiple Gaussian toy data sets are generated

⁷Since no edge is shared from the data item to the green or “square” class, we set the network measure variations regarding the insertion of the data item 1 to their maximum value (as explained in the previous section).

via PRTools ([Duin, 2000](#)). Each class is constructed with a distinct mean and spatial correlation and has 500 vertices. The error estimation is done using the stratified ten-fold cross-validation fifty times. The network construction parameters for the training phase are set to $k = 3$ and $\epsilon = 0.05$. The same k and ϵ are utilized in the classification phase. The fuzzy SVM with RBF kernel ($C = 2^2$ and $\gamma = 2^1$) is adopted as the low level classifier.

Observing all the aforementioned steps, we advance to our first experiment in which there are two completely separated classes in the space, as the scatter plot in Fig. [5.9a](#) exhibits. By a quick analysis of it, one can infer that any efficient low level classifier can achieve high accuracy rates, given the well-behaved classes. This aspect can be exactly observed in Fig. [5.9b](#) which shows the accuracy rate of the proposed technique for various values of ρ . Specifically, for $\rho = 0$ (only the usage of the traditional classifier), almost no wrong labels are assigned. As we increase the compliance term, more weight is given for the outcome of the high level classifier and, consequently, less relevance is given for the decision derived by the traditional low level classifier. By Fig. [5.9b](#), one can notice that the classification remains steady at practically 100%, provided that $\rho \leq 0.8$. For values greater than this, the classification rate starts to monotonically decrease, as ρ increases. We have checked the vertices that have been misclassified when $\rho = 1$ and have found that they are situated in their entirety in the class borders. This is predictable, since the two classes are similar and, hence, the network measures associated to each representative component are almost equivalent. Arguments such that there is no mixture between the classes and the spatial correlations of both classes are similar reinforce this phenomenon. Therefore, it is natural that the high level classifier will become confused in classifying these data items under such conditions, as the pattern formation of both classes are similar or insignificant. This example shows that only the high level classifier is insufficient to get good classification results.

Now let us examine the problem of classification illustrated in Fig. [5.9c](#), where the two classes now slightly collide with each other. As a result of this phenomenon, a conflicting region is constituted. The data items which reside in this region are most likely to be misclassified by a pure traditional classifier. Bearing in mind these considerations, let us proceed our attention to Fig. [5.9d](#), which displays the accuracy rate of the proposed technique for various values of ρ . One can see that a mixture of traditional low level and high level classifiers does supply a boost in the classification rate. Specifically, for $\rho = 0$, 74% of the vertices in average are correctly classified, whereas with a little influence of the high level classifier that we have developed, it increases to 79% in average ($\rho = 0.22$). Likewise the previous example, as ρ increases after this peek, the accuracy rate monotonically decreases by the same reasons we have stated before. The overlapping region of both classes impacts the network construction. In this way, the representative class components become slightly different. These argu-

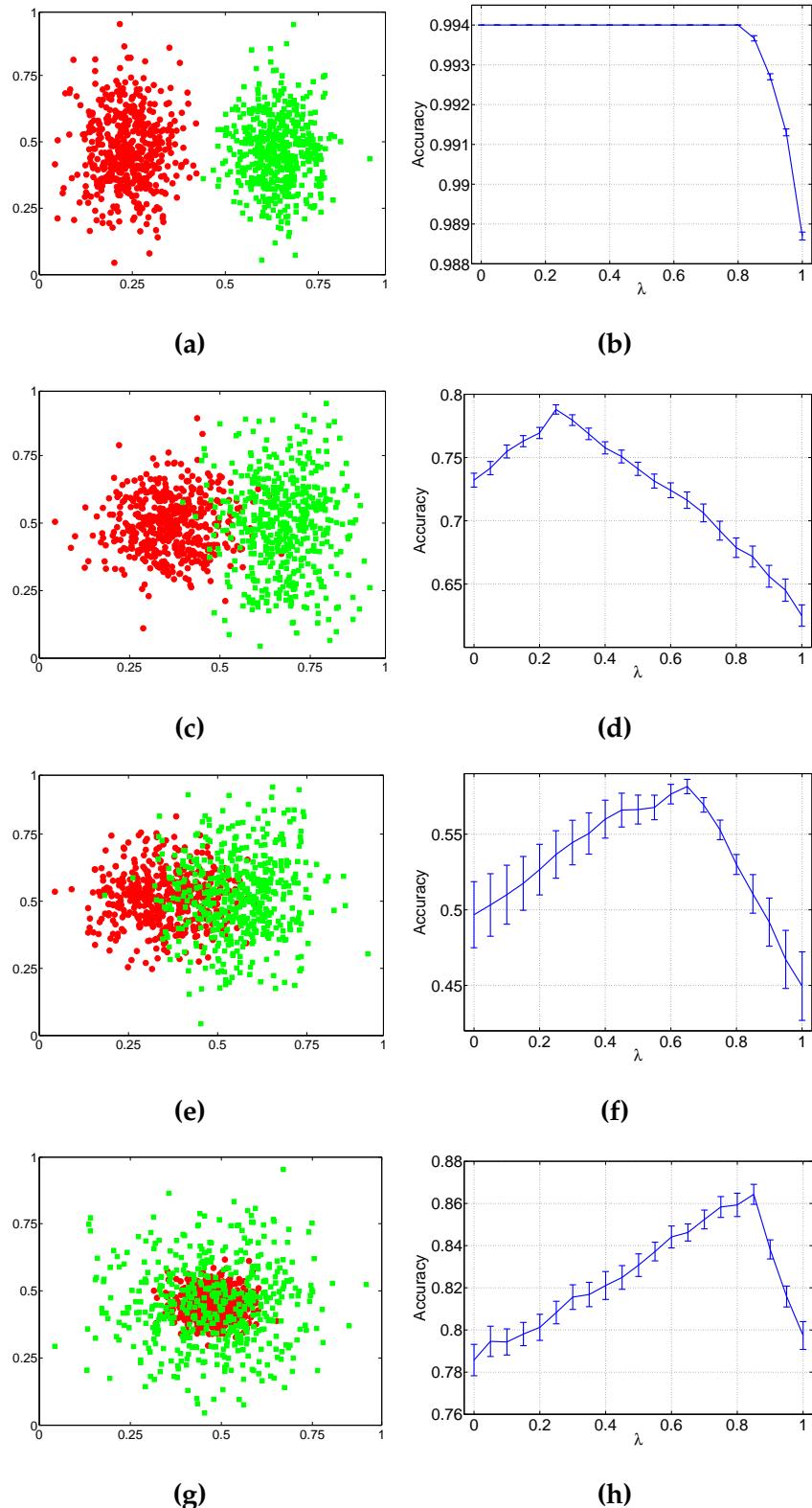


Figure 5.9: Impact analysis of the compliance term on the model's accuracy rate when Gaussian distributions are employed. (a) Two integrally separated classes. (b) Impact of ρ on the network of Fig. 5.9a. (c) Two classes slightly mixed. (d) Impact of ρ on the network of Fig. 5.9c. (e) Two classes heavily mixed. (f) Impact of ρ on the network of Fig. 5.9e. (g) Two classes, one with data items condensed in a small area and another class with data items scattered throughout a significant big area. (h) Impact of ρ on the network of Fig. 5.9g.

ments explain that a little relevance to the high level classifier decision, in this specific case, can cause the classification rate to increase.

We now proceed to inspect the classification problem proposed in Fig. 5.9e. In this particular situation, the two classes heavily collide with each other. Due to that, the two classes become almost indistinguishable. In consequence of the network formation technique, two very distinct and representative components will arise from these two classes. In this special scenario, the two components that comprise the network are expected to possess different network properties, i.e., some structural pattern is hoped for emerging. Indeed, taking a careful view in Fig. 5.9f, which depicts the accuracy rate reached against various values of ρ , one can state that the accuracy rate keeps increasing until a high value of the compliance term, namely $\rho = 0.62$, where it achieves 53%, against 49% when $\rho = 0$. This phenomenon is precisely explained by the structural patterns that each component provides, in which merely traditional classifiers would not be able to capture. Therefore, a heavy weight on the high level classifier decision is responsible for this significant increase in the classification rate.

As our last Gaussian distribution problem, consider Fig. 5.9g, where, in this case, the correlation matrix of the green or "square" class is thrice the correlation matrix of the red or "circle" class, i.e., the former becomes more scattered than the latter. This supplies a natural way of building class patterns, where the average physical distance among the data contributes to the definition of the class. Additionally, this is reinforced by the way the network is formed. In fact, Fig. 5.9h, which illustrates the accuracy rate reached against various values of ρ , precisely shows this phenomenon. Classifiers solely based on similarity measures would generally misclassify all the green or "square" vertices that are situated in the condensed region of the red or "circle" class. A high value of the compliance term ρ can overcome this problem, giving more relevance to the patterns that each component possesses and less influence on physical distances among the data. Indeed, we can observe that the highest classification rate is reached when $\rho = 0.80$ with 92% of the vertices correctly classified against 79%, when $\rho = 0$. A cautious investigation shows that for $\rho = 0$, practically all green or "square" vertices inside the red or "square" class' region are indeed misclassified, as we have stated. However, for $\rho = 0.80$, the majority of these vertices can be correctly classified.

In summary, we end this section by pointing out the key points that we have inferred from these synthetic Gaussian distributed examples:

- When the classes are strictly separated, no aid from the high level classifier is necessary, i.e., $\rho = 0$ is enough;
- When the classes slightly collide, $\rho > 0$ is required and its magnitude increases with the mixture;
- When the classes show clear patterns, $\rho = 1$ suffices. The classifier will be able

to correctly complete the patterns associated to one or more classes, even in the situations where the data items invade some other class' region.

- In general, neither the individual low level term nor the high level term by themselves produce the best accuracy rate results.

5.4.2 Simulations on Real-World Data Sets

In this section, the proposed framework is tested against several well-known UCI data sets. A succinct meta-information of the selected data sets is given in Table 5.3. For a detailed description, one can refer to ([Frank and Asuncion, 2010](#)). Concerning the numerical attributes, the used similarity measure is the reciprocal of the Euclidean distance. For the categorical examples, the overlap similarity measure is employed ([Boriah et al., 2008](#)). All predictive attributes are standardized such as to present zero mean and unitary standard deviation.

Table 5.3: Meta-information of the data sets.

	# Samples	# Dimensions	# Classes
Yeast	1 484	8	10
Teaching	151	5	3
Pima	768	9	2
Zoo	101	16	7
Wine	178	13	3
Iris	150	4	3
Glass	214	9	6
Vehicle	846	18	4
Letter	20 000	16	26

The high level classifier is composed of a weighted combination of the three complex network measures described before, namely assortativity, clustering coefficient, and average degree. Instead of using a brute force method to find the weights of each network measure in (5.5), they are heuristically tuned with the aid of the Particle Swarm Optimization technique (PSO) ([Kennedy and Eberhart, 1995](#)). In the PSO technique, a particle swarm is modeled via particles in a multidimensional space, each of which with its own position and velocity. These particles navigate in this space (solution space) trying to find the best candidate solution with regard to a quality function. At each time step, the particle swarm optimization concept consists in changing the velocity of each particle toward the best candidate solution in the vicinity. Here, the system is initialized with a random population of particles and the optimization criterion is set to be the accuracy of the model. The accuracy is estimated using a stratified ten-fold cross-validation method. The solution space is discretized as fol-

lows: $\alpha(i) \in \{0.1, 0.2, \dots, 1.0\}, \forall i \in \{1, 2, 3\}$, subjected to $\sum_{u=1}^3 \alpha(u) = 1$. According to the rule-of-thumb indicated in (Shi and Eberhart, 1998), we set $\omega = 0.7298$, $\varphi_1 = \varphi_2 = 1.49618$ for PSO. The parameter optimization results are given in Table 5.4. Each row indicates the best weighting combination of the three complex network measures. For instance, take the Yeast data set, the high level classifier supplies the best performance when the weights of the assortativity, clustering coefficient, and average degree are 0.4, 0.4, and 0.2, respectively. The network in the training phase is constructed using $\epsilon = 0.03$ and $k = 3$. However, in the classification phase, only k is maintained as the same and ϵ is optimized over the set $\{0.01, 0.02, \dots, 0.1\}$.

Table 5.4: Weighting factors of each network measure used in the first implementation of the high level classifier.

	Assortativity ($\alpha(1)$)	Clustering Coefficient ($\alpha(2)$)	Average Degree ($\alpha(3)$)
Yeast	0.4	0.4	0.2
Teaching	0.6	0.1	0.3
Pima	0.4	0.4	0.2
Zoo	0.5	0.1	0.4
Wine	0.6	0.3	0.1
Iris	0.6	0.2	0.2
Glass	0.3	0.2	0.5
Vehicle	0.4	0.3	0.3
Letter	0.3	0.5	0.2

With respect to the low level classifiers, the following techniques are used: Bayesian networks (Neapolitan, 2003), the weighted version of the k -NN algorithm (Hastie *et al.*, 2009), a fuzzy C4.5 method (Olaru, 2003), multilayer perceptrons (MLP) (Rumelhart *et al.*, 1988), and the fuzzy multiclass SVM (M-SVM) (Abe and Inoue, 2002; Lin and Wang, 2002). Every parameter of the low level classifier will be optimized using the best combination of parameters discussed as follows. The only parameter of the Bayesian networks technique is the maximum allowed number of parent attributes that an attribute can depend on. For instance, if all attributes cannot depend on any other attributes (independence assumption), then we have the Naïve Bayes classifier. If just one parent is allowed per each attribute, we have the Boosted Augmented Naïve Bayes classifier, and so on. Here, we simply make the most generic choice: every attribute can depend on every other attribute of the data set. In the weighted k -NN, the classification process is performed by using the sum of the weights between the sample to be labeled and its k neighbors. Specifically, the weight between two samples x_i and x_j is defined as the reciprocal of the Euclidean distance. For the fuzzy C4.5 algorithm, the parameter setup is taken integrally from (Olaru, 2003). For the MLP technique, three parameters are optimized, namely (i) the number of layers in the

model, which is optimized over the set $\{1, 2, 3, 4, 5, 6, 7\}$; (ii) the learning rate, which is optimized over the interval $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, and (iii) the momentum term, whose optimization is restricted to the set $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. Lastly, regarding the fuzzy SVM algorithm, we have opted to utilize the *one-against-one* multi-class version, in which $\binom{L}{2}$ distinguished binary classifiers (every pair of classes) are trained and the decision is set by a voting scheme. In order to avoid ties, the output of each SVM correspond to real-valued decision functions (fuzzy). As Hsu draws attention to, the one-against-one SVM is preferable over others multiclass methods and yields the best results among them (Hsu and Lin, 2002). By virtue of the computational complexity of the one-against-one multiclass version of the SVM technique, we reduce the search-space for the optimization process by fixing a single well-known kernel, namely the RBF kernel, previously considered in the synthetic examples. The stopping criterion for the optimization method is defined as the Karush-Kuhn-Tucker violation to be less than 10^{-3} . For each data set, the model selection is performed by considering the kernel parameter $\gamma \in \{2^4, 2^3, 2^2, \dots, 2^{-10}\}$ and the cost parameter $C \in \{2^{12}, 2^{11}, 2^{10}, \dots, 2^{-2}\}$. We have chosen a wide range of possible values for the cost parameter because it is responsible for creating soft margins and is usually welcomed in problems where the classes are almost linearly separable (after the application of the kernel transformation). A low value creates hard margins, whereas a big value, flexible margins.

Table 5.5 reports the results of the proposed technique with the five different low level classifiers explained before. The results obtained by each algorithm are averaged over a hundred runs using the stratified ten-fold cross-validation process. For each data set, two types of results are depicted: (i) “Pure” row: shows the accuracy rate when $\rho = 0$, along with the parameters of the low level classifier inside the parentheses; (ii) “Best” row: indicates the best accuracy rate reached when we use both low and high level classifiers. In this case, inside the parentheses are reported the ϵ in the classification phase and the best compliance term ρ that achieved the highest accuracy rate. Take the first entry for instance, the pure low level classifier Bayesian Networks achieved an accuracy rate of 57.8 ± 2.6 ($\rho = 0$). However, if we utilize a $\rho = 0.21$ acting together with a high level classifier whose ϵ used in the classification phase is $\epsilon = 0.06$, the accuracy rate is refined, achieving 63.7 ± 2.2 . One can see that the proposed method can satisfactorily improve the classification accuracy rates in the majority of the cases. It should be mentioned that the results obtained by only using the traditional classification techniques are already optimized; thus, the increasing of the classification accuracy over these optimized results is in general hard to be obtained. In this sense, the proposed technique presents a significant advantage in general classification problems.

Table 5.5: Comparison results obtained by 5 well-known supervised data classification techniques on 8 distinct data sets. The accuracy rate and the standard deviations are reported. For each entry of the row named “Pure,” some parameters are supplied: Weighted k -NN (k), fuzzy C4.5, MLP (number of layers, learning rate, momentum), fuzzy M-SVM (C, γ). For each entry of the row denominated “Best,” the best ϵ in the classification phase and the best ρ are reported, respectively.

		Bayesian Networks		Weighted kNN		Fuzzy C4.5		MLP		Fuzzy M-SVM	
Yeast	Pure	57.8	± 2.6	60.9	± 3.6 (16)	57.8	± 3.9 (0.1, 5)	56.2	± 3.9 (4, 0.3, 0.2)	58.9	± 4.8 (2 ¹¹ , 2 ⁰)
	Best	63.7	± 2.2 (0.06, 0.21)	68.6	± 2.8 (0.04, 0.33)	58.3	± 3.2 (0.04, 0.15)	62.8	± 2.5 (0.06, 0.28)	70.0	± 4.3 (0.03, 0.32)
Teaching	Pure	61.3	± 8.8	63.0	± 12.3 (9)	64.6	± 14.9	60.9	± 9.4 (7, 0.2, 0.4)	52.5	± 7.9 (2 ⁶ , 2 ³)
	Best	68.3	± 7.5 (0.02, 0.24)	69.6	± 9.9 (0.03, 0.21)	67.0	± 8.4 (0.02, 0.22)	69.0	± 8.1 (0.01, 0.28)	63.8	± 4.6 (0.04, 0.37)
Pima	Pure	74.4	± 3.9	74.0	± 4.4 (7)	73.6	± 1.3	75.4	± 5.1 (3, 0.2, 0.3)	75.7	± 1.7 (2 ¹ , 2 ²)
	Best	79.7	± 3.2 (0.05, 0.15)	79.2	± 2.8 (0.03, 0.13)	75.9	± 1.5 (0.04, 0.12)	81.6	± 3.7 (0.05, 0.17)	81.9	± 1.2 (0.06, 0.18)
Zoo	Pure	95.9	± 4.3	96.2	± 5.8 (1)	96.1	± 3.2	96.1	± 6.9 (3, 0.4, 0.5)	96.3	± 6.4 (2 ¹ , 2 ¹)
	Best	97.3	± 3.4 (0.03, 0.12)	97.8	± 4.3 (0.04, 0.10)	96.1	± 3.2 (-, 0.00)	98.1	± 5.2 (0.05, 0.18)	98.4	± 3.4 (0.04, 0.14)
Wine	Pure	98.8	± 0.7	94.6	± 1.4 (1)	96.5	± 0.8	97.8	± 0.5 (3, 0.6, 0.4)	98.9	± 0.2 (2 ¹¹ , 2 ²)
	Best	98.8	± 0.7 (-, 0.00)	97.1	± 1.9 (0.03, 0.19)	97.6	± 1.0 (0.02, 0.16)	98.7	± 0.5 (0.02, 0.07)	98.9	± 0.2 (-, 0.00)
Iris	Pure	92.7	± 1.2	97.9	± 3.3 (19)	95.3	± 0.9	94.0	± 2.9 (1, 0.3, 0.2)	97.0	± 4.6 (2 ⁻² , 2 ³)
	Best	93.2	± 0.9 (0.01, 0.25)	97.9	± 3.3 (-, 0.00)	95.9	± 0.9 (0.01, 0.06)	94.6	± 3.4 (0.01, 0.14)	97.0	± 4.0 (0.01, 0.09)
Glass	Pure	70.6	± 7.7	71.8	± 9.0 (1)	71.1	± 1.5	67.3	± 5.0 (7, 0.1, 0.3)	72.4	± 5.6 (2 ¹⁰ , 2 ⁴)
	Best	78.2	± 4.9 (0.06, 0.32)	80.6	± 6.6 (0.04, 0.34)	80.3	± 4.9 (0.05, 0.34)	78.2	± 5.9 (0.04, 0.36)	79.8	± 4.4 (0.05, 0.26)
Vehicle	Pure	68.1	± 3.8	67.6	± 4.1 (5)	70.1	± 2.7	69.0	± 4.4 (5, 0.3, 0.2)	84.4	± 3.4 (2 ¹⁰ , 2 ³)
	Best	75.8	± 4.6 (0.03, 0.34)	76.2	± 3.5 (0.05, 0.37)	75.8	± 2.4 (0.03, 0.26)	75.1	± 2.7 (0.04, 0.30)	85.9	± 2.6 (0.03, 0.12)
Letter	Pure	74.4	± 5.6	96.0	± 7.6 (1)	88.9	± 8.1	88.9	± 9.9 (3, 0.2, 0.4)	94.8	± 1.7 (2 ⁶ , 2 ⁴)
	Best	84.7	± 5.5 (0.06, 0.22)	96.0	± 7.6 (-, 0.00)	88.9	± 8.1 (-, 0.00)	92.3	± 8.7 (0.07, 0.09)	96.0	± 1.1 (0.05, 0.08)

5.4.3 Application: Handwritten Digits Recognition

In this section, we show that the high level learning provided by the proposed technique can indeed help in solving real-world problems, specifically those pertaining to the Pattern Recognition area. In this context, the proposed technique is applied to the recognition of handwritten digits, such as those available from the MNIST data set.

It is worth registering that we have employed the same weighted eigenvalue-based network formation technique as in Section 3.5, when we dealt with the same application, but in another perspective, which is clustering, rather than recognition.

Description of the MNIST Data Set

Handwriting digits recognition is a well accepted benchmark for comparing pattern recognition methods. In this study, our model is applied to the Modified NIST (MNIST) data set (LeCun *et al.*, 1998), which comprises a training set of 60 000 samples and a test set of 10 000 samples. This data set is almost balanced with regard to the size of the 10 existing classes, each of which representing a digit. Similarly to LeCun *et al.* (1998), a pre-processing step is conducted. In this respect, the gray-level images (samples) are reduced to fit in a 20×20 pixel box, while preserving their aspect ratio.

Configurations of the Low Level Classifiers

Below, we detail the three low level classifiers that are going to be used in the following computer simulations. For more details about the setup and architectural characteristics of the first two techniques, one can refer to (LeCun *et al.*, 1998).

- *A Perceptron neural network:* each input pixel value contributes to a weighted sum for each output neuron. The output neuron with the highest sum (including the contribution by virtue of the bias applied to that neuron) marks the class of the input character.
- *A k-nearest neighbor classifier:* we set the similarity as the reciprocal of the Euclidean distance and $k = 3$.
- *A network-based ϵ -radius classifier:* the ϵ -radius network formation technique is employed with a dissimilarity measure given by a weighted sum of the $\phi = 4$ greatest eigenvalues, as described in the previous section. In all simulations, $k = 3$ and $\epsilon = 0.014$ for the both steps (classification and network formation).

Experimental Results

Figure 5.10 shows the accuracy rate reached by the three discussed techniques against several values of ρ on the MNIST data set. Our main goal here is to reveal

that a proper mixture of traditional and high level classifiers is able to increase the overall accuracy rate of the predicting model. Investigating this figure, one can see that the perceptron neural network achieved 88% of classification rate when only a traditional classifier ($\rho = 0$) is employed. A little increase of the compliance term, in the case, $\rho = 0.25$, is responsible for a boost in the accuracy rate. In particular, it achieves almost 91% of correctly labeled test instances. With regard to the k -nearest neighbor algorithm, for a pure traditional classifier, we have obtained 95% of accuracy rate, against 96% when $\rho = 0.2$. In the ϵ -radius classifier, we have obtained 98.49% of correctly classified data items for $\rho = 0$, against 99.06% when $\rho = 0.2$. Finally, this experiment indicates that the optimal compliance term might be intrinsic to the data set, since, for three completely distinct low level classifiers, the maximum accuracy rate is achieved in the surroundings of $\rho = 0.225$. A proof of this conjecture is left as future work.

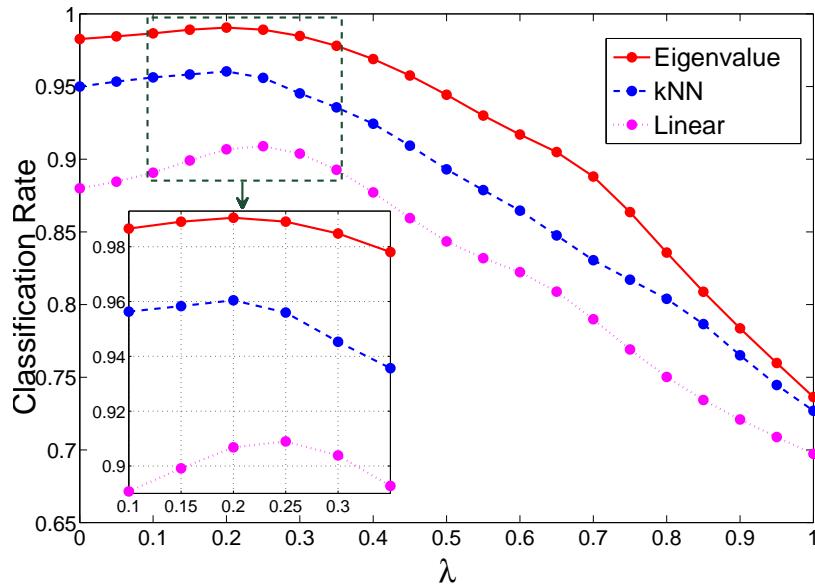


Figure 5.10: An analysis of the impact of the compliance term ρ on three different traditional low level techniques applied to the MNIST database.

Table 5.6 reports two samples (the first two rows) that are easily classified and four samples (the last four rows) that are hard to be correctly classified by only using traditional techniques. Each entry of the Estimated Membership Values section is composed of two values: the top-most is the fuzzy output of the linear classifier and the bottom-most is the fuzzy output of the high level classifier. The Final Prediction section of this table indicates the outcome when only the linear classifier is applied ($\rho = 0$) and when a mixture of it and the high level classifier is employed ($\rho = 0.2$). For the sake of completeness, the samples shown in this table have been taken from the test set of the MNIST data set and have the following IDs, from the top to the bottom-most: 52, 54,

5594, 7171, 4301, and 7427. From these classification results, we see that the proposed technique is able to correctly identify patterns with considerable distortions. This is because the high level classification examines not only the similarity between the test digit and the training digits (training set), but also exploits the digits presenting intermediate variations to form a pattern for each digit via the networked representation of the data.

5.5 Computer Simulations using the High Level Classifier based on Tourist Walks

In this section, we assess the performance of the high level classifier composed by the dynamical information generated from the tourist walks. Specifically, Section 5.5.1 provides simulations on artificial data sets in order to better motivate the proposed model; Section 5.5.2 supplies a parameter sensitivity analysis of the model; Section 5.5.3 presents test results obtained from real-world data sets; and Section 5.5.4 explores a real-world recognition task. The error estimation method in these simulations is set to be the stratified ten-fold cross-validation.

5.5.1 Illustrative Examples

For the sake of clarity, in this section, the weights given for the transient and cycle lengths are the same, i.e., $\alpha_t = \alpha_c = 0.5$. Moreover, the critical tourist walk length μ_c is set as being the size of the smallest class in the problem. As before, here we will mainly focus on particular situations where low level classifiers would have trouble in correctly classifying the data items in the test set.

As an introductory example, we revisit the networked data set shown in Fig. 5.11, which comprises 2 classes, a red or "circle" (35 vertices) class and a blue or "square" (160 vertices) class. In view of that, the proportion of vertices pertaining to each class is: $p^{(\text{red})} = 17.95\%$ and $p^{(\text{blue})} = 82.05\%$. As we have deducted in the previous sections, while the red ("circle") class carries a perceivable pattern, which, geometrically speaking, is a grid or lattice disposed in a two-dimensional space, the blue ("square") class does not indicate any clear patterns. Again, in this particular situation, we will see that the results produced by the SVM by itself are not satisfactory, since it predicts using traditional assumptions of the data. In this case, the fuzzy SVM attempts to push the support vectors away from the large amounts of green or "square" data items in the surroundings.

Table 5.7 reports the classification results of the test instances for all the 4 kernels. For each kernel, the final prediction is given for three different values of the compliance term ρ : $\rho = 0$ (pure SVM), $\rho = 0.5$ (equal weight for the decision of the SVM and the

Table 5.6: Classification membership values of the low level and high level classifiers.

Sample	Real Class	Estimated Membership Values (Classes)									Final Prediction	
		0	1	2	3	4	5	6	7	8	9	$\rho = 0$
	Class 3	0.11	0.00	0.00	0.74	0.00	0.00	0.00	0.15	0.00	0.00	Class 3
	Class 3	0.23	0.00	0.00	0.57	0.00	0.00	0.00	0.20	0.00	0.00	Class 3
	Class 5	0.00	0.00	0.00	0.00	0.79	0.16	0.00	0.05	0.00	0.00	Class 5
	Class 5	0.00	0.00	0.00	0.00	0.47	0.33	0.00	0.20	0.00	0.00	Class 5
	Class 0	0.39	0.00	0.00	0.00	0.00	0.43	0.01	0.12	0.05	0.05	Class 6
	Class 0	0.54	0.00	0.00	0.00	0.00	0.31	0.00	0.10	0.05	0.05	Class 0
	Class 3	0.02	0.00	0.00	0.40	0.00	0.00	0.44	0.11	0.03	0.03	Class 7
	Class 3	0.07	0.00	0.00	0.54	0.00	0.00	0.32	0.05	0.02	0.02	Class 3
	Class 5	0.15	0.00	0.00	0.00	0.34	0.39	0.00	0.05	0.07	0.07	Class 6
	Class 5	0.06	0.00	0.00	0.00	0.51	0.36	0.00	0.07	0.00	0.07	Class 5
	Class 9	0.20	0.02	0.00	0.00	0.27	0.14	0.00	0.13	0.00	0.24	Class 4
	Class 9	0.00	0.00	0.00	0.00	0.17	0.41	0.00	0.05	0.00	0.37	Class 9

Table 5.7: Results obtained for the classification of the triangle-shaped data items in Fig. 5.11. The fuzzy SVM technique has been used for the low level classifier. The first item is the top-most triangle-shaped item and the fifth item is the bottom-most triangle-shaped item.

	Kernel	Optimized Parameters			$\rho = 0$		$\rho = 0.5$		$\rho = 0.9$		
		C	γ	c	d	Red Class	Blue Class	Red Class	Blue Class	Red Class	Blue Class
First	Linear	2^1	N/A	N/A	N/A	0.40	0.60	0.54	0.46	0.64	0.36
	RBF	2^0	2^{-6}	N/A	N/A	0.47	0.53	0.60	0.40	0.67	0.33
	Sigmoid	2^0	2^{-1}	-4.5	N/A	0.46	0.54	0.59	0.41	0.67	0.33
	Polynomial	2^1	2^{-4}	3.0	4	0.44	0.56	0.57	0.44	0.65	0.35
Second	Linear	2^{-1}	N/A	N/A	N/A	0.24	0.76	0.40	0.60	0.51	0.49
	RBF	2^{-1}	2^{-5}	N/A	N/A	0.29	0.71	0.45	0.55	0.57	0.43
	Sigmoid	2^{-1}	2^{-3}	-6.5	N/A	0.31	0.69	0.47	0.53	0.58	0.42
	Polynomial	2^{-1}	2^{-5}	5.0	2	0.28	0.72	0.44	0.56	0.57	0.43
Third	Linear	2^4	N/A	N/A	N/A	0.33	0.67	0.47	0.53	0.59	0.41
	RBF	2^6	2^{-3}	N/A	N/A	0.39	0.61	0.51	0.49	0.62	0.48
	Sigmoid	2^4	2^1	-3.5	N/A	0.36	0.64	0.49	0.51	0.60	0.40
	Polynomial	2^5	2^{-1}	-0.5	9	0.36	0.64	0.49	0.51	0.60	0.40
Fourth	Linear	2^2	N/A	N/A	N/A	0.41	0.59	0.57	0.43	0.65	0.35
	RBF	2^2	2^{-4}	N/A	N/A	0.49	0.51	0.62	0.38	0.70	0.30
	Sigmoid	2^3	2^{-2}	-2.5	N/A	0.49	0.51	0.62	0.38	0.70	0.30
	Polynomial	2^2	2^{-2}	1.0	7	0.42	0.58	0.58	0.41	0.65	0.35
Fifth	Linear	2^{-1}	N/A	N/A	N/A	0.22	0.78	0.39	0.71	0.51	0.50
	RBF	2^0	2^{-6}	N/A	N/A	0.26	0.74	0.42	0.58	0.55	0.45
	Sigmoid	2^0	2^{-1}	-4.5	N/A	0.26	0.74	0.42	0.58	0.55	0.45
	Polynomial	2^0	2^{-4}	4.0	4	0.25	0.75	0.40	0.60	0.52	0.48

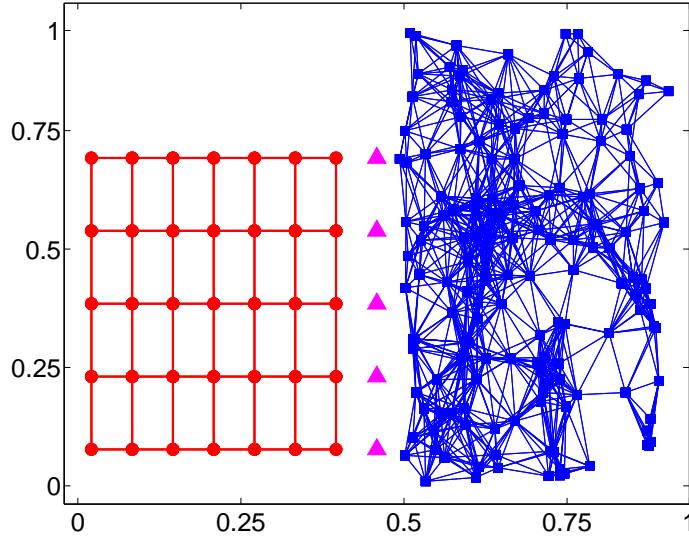


Figure 5.11: A high level data classification task. A well-structured and organized two-dimensional lattice (grid) class (henceforth, referred to as “circle” or red class) and a condensed opaque squared class (called “square” or blue class).

high level classifier), and $\rho = 0.9$ (a high weight for the high level classifier and a low weight for the SVM). We can clearly see that a pure SVM with well-known kernels is not able to correctly classify the data items shown in Fig. 5.11. However, if we make our final decision based on a mixture of the SVM and the high level classifier, one can verify that correct results can be obtained. Except for the Linear kernel, for which the algorithm derives a wrong classification decision even with the help of the high level classifier ($\rho = 0.5$), we can conclude from the table that all other kernels are capable of supplying the correct answer when $\rho \in \{0.5, 0.9\}$.

Now, consider the classification problem arranged in Fig. 5.12. Here, we are going to empirically calculate the minimum required compliance term ρ_{\min} for which the data items from the test set are classified as members of the red or “circle” class. In the figure, one can see that there is a segment of line representing the red or “circle” class (9 vertices) and also a condensed rectangular class outlined by the blue or “square” class (1000 vertices). The network formation in the training and classification phases use $k = 1$ and $\epsilon = 0.07$. The fuzzy SVM technique with RBF kernel ($C = 2^2$ and $\gamma = 2^{-1}$) is employed as the traditional low level classifier. The task is to classify the 14 test instances depicted by the big triangle-shape items from left to right. After a test instance is classified, it is incorporated to the training set with the corresponding predicted label. The graphic embedded in Fig. 5.12 shows the minimum required value of ρ_{\min} for which the triangle-shaped items are classified as members of the red or “circle” class. This graphic is constructed according to the triangle-shaped element

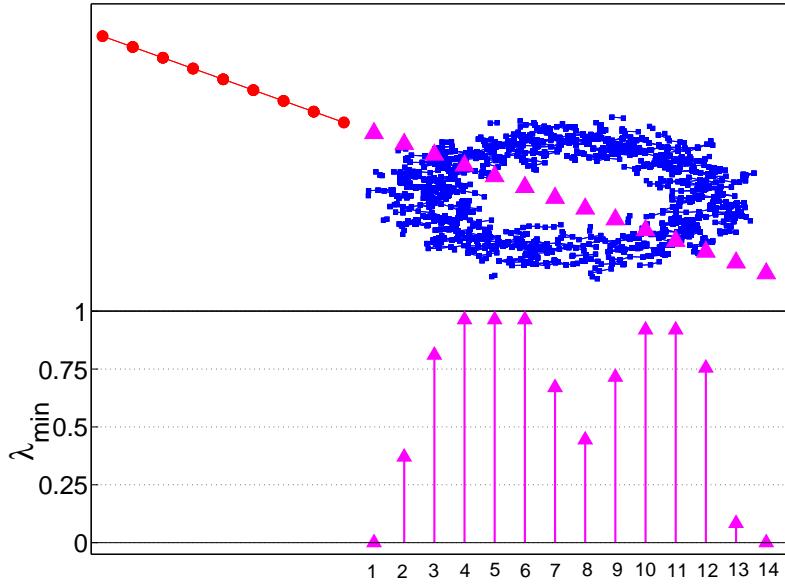


Figure 5.12: Minimum value of the compliance term, ρ_{\min} , that results in the correct classification of the missing test instances. Traditional techniques would definitely fail to correctly classify the straight line that diagonally crosses the densely connected component pertaining to the blue or “square” class.

that is exactly at the same position with respect to the x -axis in the scatter plot drawn above. For example, the second triangle-shaped data item can be correctly classified if one chooses $\rho \geq \rho_{\min} \approx 0.37$. The third and forth data items would require at least $\rho_{\min} = 0.81$ and $\rho_{\min} = 0.96$, respectively, and so on. Specifically, as the straight line crosses the condensed region of the blue or “square” class, $\rho \rightarrow 1$, since one cannot establish its decision based on the low level classifier, because it would erroneously decide favorable to the blue or “square” class.

5.5.2 Parameter Sensitivity Analysis

In this section, we will perform several simulations with the goal of better understanding the influence of each parameter in the model.

Influence of the Compliance Term for Different High Level Classifiers

In this section, we study the influence of the compliance term ρ for three different types of high level classifiers, as follows: (i) one constructed solely using the component’s cycle length; (ii) one built exclusively using the component’s transient length; and (iii) the best weighted combination of these two measures. The optimization process is conducted by finding $\alpha_t \times \alpha_c \in \{0, 0.1, \dots, 1\} \times \{0, 0.1, \dots, 1\}$ (search space), subjected to $\alpha_t + \alpha_c = 1$, which results in the highest accuracy rate of the model. Here, we show that, in general, the transient and cycles lengths are not correlated. The ob-

jective of this section is to make this point clear and demonstrate that, when they act together, they are able to produce better accuracy rates. In order to conduct these tests, classes with Gaussian distributions are used, as we have seen before. The network in the classification phase is constructed with $k = 1$ and $\epsilon = 0.05$. The same configuration is employed in the classification phase. The similarity measure is simply given by the reciprocal of the Euclidean measure. For the traditional classifier, the fuzzy SVM with RBF kernel is employed with $C = 2^4$ and $\gamma = 2^{-2}$. Finally, this process is repeated 100 times and the mean and the corresponding standard deviation of the accuracy rate, for each value of ρ , is reported.

Consider Fig. 5.13a, where the classes are completely separated from each other. Figure 5.13b depicts the accuracy rate of the proposed classifier vs. ρ for the three types of high level classifiers discussed before. The best weighted combination of the cycle and transient lengths is $\alpha_c = 0.6$ and $\alpha_t = 0.4$. One can see that the high level classifier constructed by the combination of the transient and cycle lengths is able to outperform the other high level classifiers constructed using only one of these measures. In addition, when $\rho = 0$ (only the usage of the traditional classifier), almost no wrong labels are assigned. On the other hand, as ρ increases, the accuracy rate of the proposed technique starts to monotonically decrease. This is predictable, since the two classes are similar; hence, the network measures associated to each representative component are almost equivalent.

Next, let us examine the problem of classification illustrated in Fig. 5.13c, where the two classes slightly collide with each other. As a result of this phenomenon, a conflicting region is constituted. The data items which reside in this region are most likely to be misclassified by a pure traditional classifier. Similarly to the previous case, Fig. 5.13d which displays the accuracy rate of the model. In this case, the best weighted combination of transient and cycle lengths is $\alpha_t = 0.7$ and $\alpha_c = 0.3$. One can see that a mixture of low level and high level classifiers does supply a boost in the accuracy rate of the model. Specifically, the highest accuracy rate occurs when $\rho \in \{0.15, 0.2\}$. The region of coalescence of both classes influences in the network construction, in such a way that the representative components of each class become slightly different. These arguments explain that a little relevance to the high level classifier decision can cause the accuracy rate to increase, since the network measures describing each component are slightly distinct.

We now proceed to inspect the classification problem proposed in Fig. 5.13e. In this particular situation, the two classes heavily collide with each other. Due to that, the two classes become almost indistinguishable. In consequence of the network formation technique, two very distinct and representative components will arise from these two classes. In this special scenario, the two components that comprise the network are expected to be possess different network properties, i.e., some unique structural

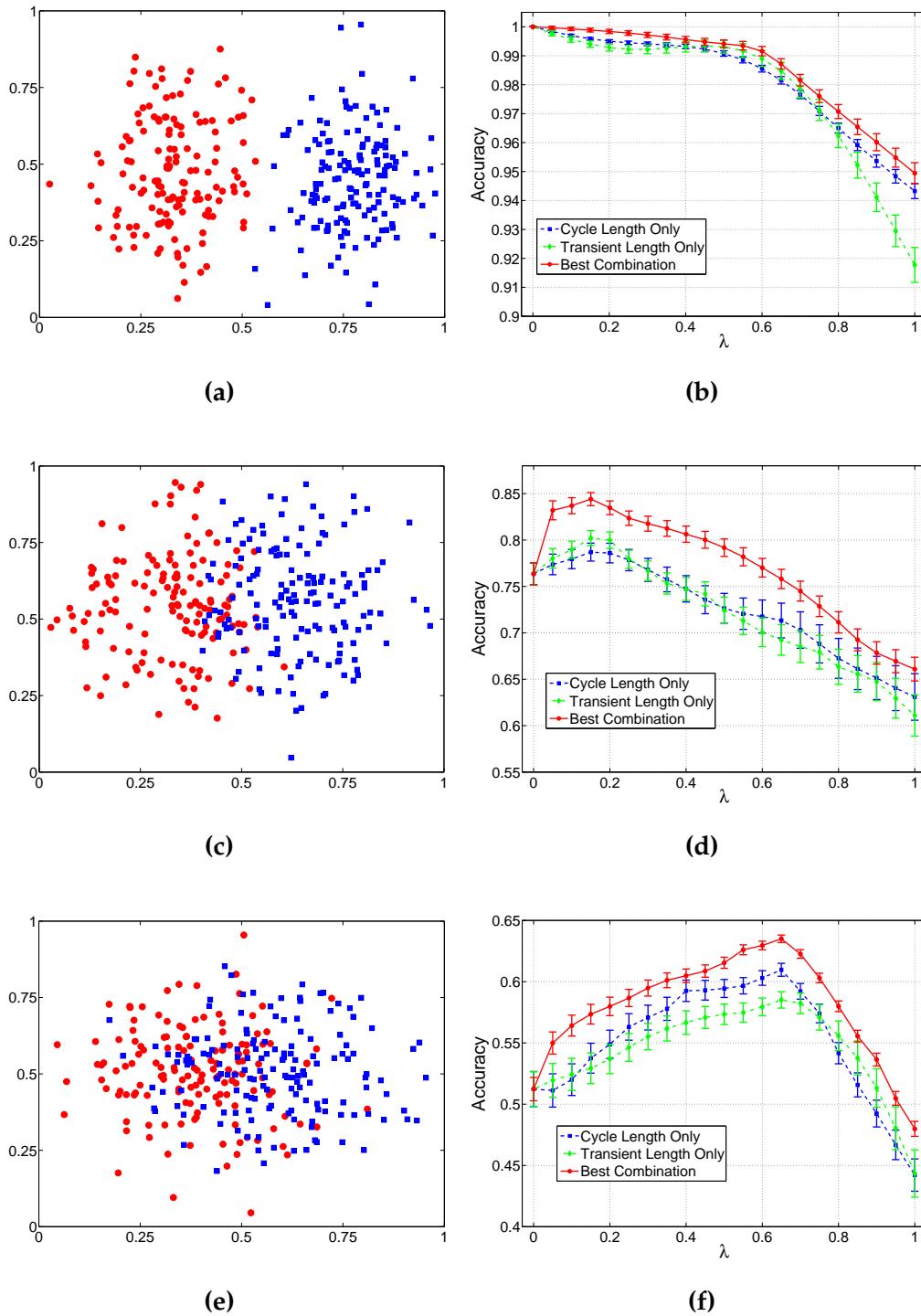


Figure 5.13: Analysis of accuracy rate vs. the compliance term with three distinct high level classifiers. (a) Two integrally separated classes. (b) Impact of ρ on the network of Fig. 5.13a. (c) Two classes slightly mixed. (d) Impact of ρ on the network of Fig. 5.13c. (e) Two classes heavily mixed. (f) Impact of ρ on the network of Fig. 5.13e.

pattern for each class is hoped for emerging. Figure 5.13f depicts the accuracy rate reached by the classifier against various values of ρ . The best weighted combination here is $\alpha_t = 0.2$ and $\alpha_c = 0.8$. One can see that the accuracy rate keeps on increasing

until a high value of the compliance term, namely $\rho = 0.65$, where it achieves 65%, against 54% when $\rho = 0$ for the high level classifier consisted of a combination of the transition and cycle lengths.

In view of these scenarios, we can conclude that the high level classifier comprising the combination of the transient and cycle lengths is better than the high level classifier consisted of only one of these measures acting in isolation. This confirms our hypothesis that they are orthogonal and, hence, do not correlate with each other.

Influence of the Critical Memory Length

As we have seen, the high level classifier makes its prediction by using combinations of several tourist walks with increasing memory lengths: $\{0, 1, \dots, \mu_c\}$. A natural question that arises is: is it really necessary to perform tourist walks with μ ranging from 0 to the number of vertices in a component? In this section, we will empirically show that usually it is not necessary to perform all these computations.

For the tests performed in this section, we will use the well-known Iris data set from the UCI Machine Learning Repository. This data set is composed of 3 classes, each of which with 50 samples. For the traditional classifier, the fuzzy SVM with RBF kernel will be employed ($C = 2^{-2}$ and $\gamma = 2^3$). The network in the training and classification phases is constructed using $k = 1$ and $\epsilon = 0.03$. For the weight parameters of the high level classifier, the highest accuracy happens when $\alpha_c = 0.6$ and $\alpha_t = 0.4$.

Let us first understand why a smaller value of μ_c is sufficient, i.e., we do not need to fix it as the class component's size. Consider Figs. 5.14b and 5.14c, where it is depicted the transient and cycle lengths of the Iris data set. With respect to the transient length behavior, we can see that, as μ increases, the transient length also increases. However, when μ is sufficiently large, the components' transient lengths settle down in a flat region. On the other hand, the cycle length behavior is rather interesting, which can be roughly divided in three different regions: (i) for a small μ , it is directly proportional to μ ; (ii) for intermediate values of μ , it is inversely proportional to μ ; and (iii) for sufficiently large values of μ , it also settles down in a steady region. One can interpret these results as follows:

- When μ is small, it is very likely that the transient and cycle parts will also be small, because the memory of the tourist is very limited. We can conceive this as a walk with almost no restrictions;
- When μ assumes an intermediate value, the transient length keeps increasing but the cycle length reaches a peak and starts to decrease afterwards. This peak characterizes the topological complexity of the component and varies from one to another. Hence, this is the most important region for capturing pattern formation of the class component by using the topological structure of the network.

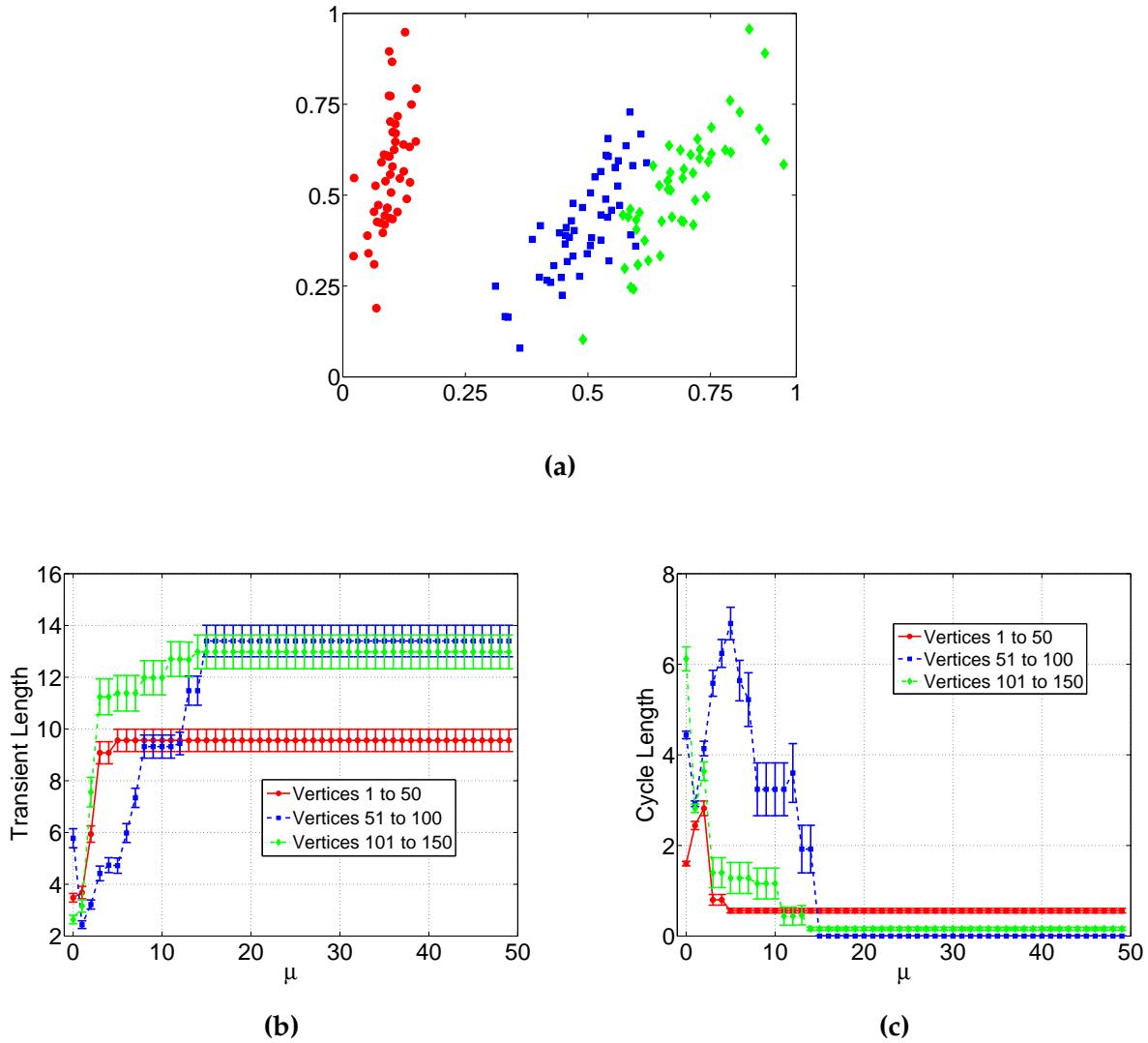


Figure 5.14: Behavior of the transient and cycle lengths of the Iris data set. (a) Scatter plot of the Iris data set using a two-dimensional PCA reduction. (b) Transient lengths vs. μ . (c) Cycle lengths vs. μ .

- When μ is large, the tourist has a greater chance of getting trapped in a vertex of the graph, once all the neighborhood of the visited vertex is contained in the memory window μ . In this scenario, the transient length is expected to be very high and the cycle length, null. This phenomenon explains the steady regions in Figs. 5.14b and 5.14c. In this region, we can say that the tourist walks have already covered all the global aspects of the class component, and increasing the memory length μ will not capture any new topological features or pattern formations of the class. In this scenario, it is said that the tourist walks have completely described the topological complexity of the class component (saturation). In view of this, the calculation of tourist walks is redundant.

This analysis suggests that the accuracy of the high level classifier will not change

given that we choose a μ_c residing near these steady regions. In order to check this, Fig. 5.15 reveals the behavior of the high level classifier for different values of μ_c . We have used three distinct compliance terms, namely $\rho \in \{0, 0.05, 0.6\}$. When $\rho = 0$, only the low level classifier is used, in such a way that the value of μ_c is irrelevant, since the high level term is disabled. With respect to the other cases, when $\mu_c \geq 20$, the model provides the same accuracy rates, confirming our prediction that, once it reaches the steady region of the transient and cycle lengths, subsequent increases of μ_c will not change the accuracy of the model. In practical terms, fixing the critical length μ_c as 10 – 30% of the component's size is enough to get satisfactory results.

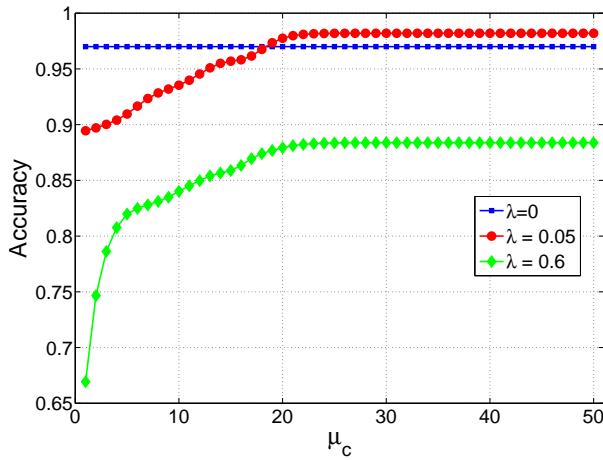


Figure 5.15: Accuracy rate vs. μ_c for 3 different values of the compliance term ρ . The mean of a stratified ten-fold cross validation is reported.

5.5.3 Simulations on Real-World Data Sets

In this section, we will apply the proposed framework to the well-known UCI data sets that we have seen in the previous section (see Table 5.3). It is worth stating that the configurations of the low level classifiers are unaltered in relation to Section 5.4.2.

Here, the high level classifier is composed of the best weighted combination of transient and cycle lengths. The optimization process is done by encountering $\alpha_t \times \alpha_c \in \{0, 0.1, \dots, 1\} \times \{0, 0.1, \dots, 1\}$ (search space), subjected to $\alpha_t + \alpha_c = 1$, which result in the highest accuracy rate of the model. The critical memory length is fixed to $\mu_c = 0.3n_{\max}$, where n_{\max} indicates the size of the largest component. The parameter optimization results are given in Table 5.8. Each row indicates the best weighting combination of the transient and cycle lengths.

Here, we will deal with two kinds of high level classifiers: (i) one in which the tourist walks are performed in a *network* constructed from the vector-based data set and (ii) one in which the tourist walks are realized in a *lattice*, i.e., the tourist is free to visit any other data site apart from the ones in the memory window μ . In the latter

Table 5.8: Weighting factors for the transient and cycle lengths used to train the high level classifier.

Data Set	Transient Length	Cycle Length
Yeast	0.40	0.60
Teaching	0.50	0.50
Zoo	0.30	0.70
Wine	0.40	0.60
Iris	0.40	0.60
Glass	0.50	0.50
Vehicle	0.60	0.70
Letter	0.40	0.60

case, it is expected that the walker will perform long jumps in the data set once the memory length μ assumes large values. It will be verified that such mechanism is not very welcomed in classification tasks. Furthermore, this serves as a strong argument for the employment and introduction of network-based high level classifiers.

Table 5.9 reports the results obtained by the proposed technique on the data sets listed in Table 5.3. The results of each algorithm are estimated over an average of a hundred runs using a stratified ten-fold cross-validation process. Also, for each result, three different types of results are indicated as follows:

- “Pure” row: only the low level classifier is utilized ($\rho = 0$). In this case, inside the parentheses are indicated the best parameters obtained from the optimization process for each technique;
- “Networkless” row: a mixture of low and high level classifiers is employed. The value inside the parentheses indicates the best compliance term ρ . The high level classifier is constructed using the best weighted combination of transient and cycle lengths with the weights respecting Table 5.8. Moreover, the tourist walks are performed in a networkless environment, i.e., the tourist can visit any other site (data item) apart from those contained in the memory window μ .
- “Network” row: the same setup as before, but the tourist walks are conducted on a networked environment. In this case, the tourist can only visit vertices (items) that are in the neighborhood and not in the memory window μ . Here, the network in the training phase is built using $k = 1$ and $\epsilon = 0.03$. The values inside the parentheses exhibit: the ϵ used in the classification phase and the best compliance term ρ , respectively.

For the sake of clarity, take the first entry of Table 5.9. The pure low level classifier Bayesian Networks achieved an accuracy rate of 57.8 ± 2.6 ($\rho = 0$). However, if we

use the proposed technique in a networkless environment, the accuracy rate is refined, achieving 58.6 ± 2.3 when $\rho = 0.04$. Now, when the proposed technique is used in a network environment, the accuracy rate reaches 66.3 ± 2.6 when $\rho = 0.28$. In general, the proposed technique is able to boost the accuracy rates of the data sets under analysis. Furthermore, we can see that the networked high level classifier can outperform the networkless version.

5.5.4 Application: Handwritten Digits Recognition

In this section, we revisit the application of handwritten digits recognition to assess the performance of the high level based on tourist walks. The same data set will be used, which is the MNIST. Additionally, here we will illustrate a pattern-based network and show how the mechanism of the high level works. We construct the network by gathering some handwritten digits from the MNIST data set. All the low level classifiers' configurations remain the same for simplicity matters.

Figure 5.16 shows the performance of these techniques acting together with the high level classifier in a networked environment. Our main goal here is to reveal that a mixture of traditional and high level classifiers is able to improve the accuracy rate. For example, the linear neural network reached 88% of accuracy rate when only a traditional classifier is applied. A small increase in the compliance term is responsible for increasing the accuracy rate to 91% ($\rho = 0.2$). Regarding the k -nearest neighbor algorithm, for a pure traditional classifier, we have obtained 95% of accuracy rate, against 97.6% ($\rho = 0.25$). For the proposed weighted eigenvalue measure, we have obtained 98% of accuracy rate when $\rho = 0$, against 99.1% when $\rho = 0.2$. It is worth noting that the enhancement is significant. Even in third case, the improvement is quite welcomed, because it is a hard task to increase an already very high accuracy rate. Moreover, one can see that maximum compliance term is intrinsic to the data set, since, for three completely distinct low level classifiers, the maximum accuracy rate is reached in the surroundings of $\rho = 0.225$.

Figures 5.17a and 5.17b illustrate how the digit classification is carried out by using simple networks containing a small number of digits '5' and '6', randomly drawn from the MNIST data set. Firstly, let us consider the Fig. 5.17a, where the digits '5' with brown boxes and the digits '6' with blue boxes represent the training set. The task now is to classify the test instance represented by the digit with a red box. If only the low level classification is applied, the test digit is probably to be classified as a digit '6', because it has more neighbors of digit '6' than that of '5'. On the other hand, if we also consider high level classification, the test digit can be correctly classified as a digit '5', because it complies more with the pattern formed by the training digits '5' than to the one formed by digits '6'. More specifically, if the test digit is put into the class '5',

Table 5.9: Accuracy rate achieved by several low level classification techniques and the high level classifier with and without networks. In the row named “Pure,” the optimized parameters of each low level technique are reported as follows: Weighted k -NN (k), MLP (number of layers, learning rate, momentum), and fuzzy M-SVM (C, γ).

Data Set	Type	Bayesian Networks	Weighted KNN	MLP	Fuzzy M-SVM
Yeast	Pure	57.8 ± 2.6	$60.9 \pm 3.6 (16)$	$56.2 \pm 3.9 (4, 0.3, 0.2)$	$58.9 \pm 4.8 (2^{11}, 2^0)$
	Networkless	$58.6 \pm 2.3 (0.04)$	$62.0 \pm 3.2 (0.07)$	$56.9 \pm 2.5 (0.05)$	$60.2 \pm 4.6 (0.14)$
Teaching Network	Pure	$66.3 \pm 2.6 (0.05, 0.28)$	$65.7 \pm 4.0 (0.03, 0.19)$	$63.3 \pm 2.9 (0.05, 0.23)$	$69.8 \pm 3.8 (0.05, 0.28)$
	Networkless	$63.5 \pm 9.3 (0.18)$	$63.8 \pm 10.6 (0.12)$	$62.0 \pm 7.7 (0.13)$	$55.3 \pm 8.6 (0.18)$
Zoo	Pure	61.3 ± 8.8	$63.0 \pm 12.3 (9)$	$60.9 \pm 9.4 (7, 0.2, 0.4)$	$52.5 \pm 7.9 (2^6, 2^3)$
	Networkless	$67.2 \pm 6.6 (0.03, 0.24)$	$68.5 \pm 7.4 (0.04, 0.19)$	$67.8 \pm 6.1 (0.04, 0.26)$	$62.7 \pm 6.9 (0.04, 0.33)$
Wine	Pure	95.9 ± 4.3	$96.2 \pm 5.8 (1)$	$96.1 \pm 6.9 (3, 0.4, 0.5)$	$96.3 \pm 6.4 (2^1, 2^1)$
	Networkless	$96.0 \pm 3.6 (0.02)$	$96.5 \pm 5.2 (0.04)$	$96.4 \pm 6.6 (0.04)$	$96.5 \pm 4.5 (0.05)$
Iris	Pure	$97.3 \pm 4.3 (0.02, 0.06)$	$97.5 \pm 4.4 (0.02, 0.09)$	$97.5 \pm 4.2 (0.02, 0.09)$	$97.5 \pm 2.3 (0.02, 0.08)$
	Networkless	98.8 ± 0.7	$94.6 \pm 1.4 (1)$	$97.8 \pm 0.5 (3, 0.6, 0.4)$	$98.9 \pm 0.2 (2^{11}, 2^2)$
Glass	Pure	$98.8 \pm 0.7 (-, 0.00)$	$94.6 \pm 2.1 (0.00)$	$97.9 \pm 0.3 (0.03)$	$98.9 \pm 0.2 (0.00)$
	Networkless	$98.8 \pm 0.7 (-, 0.00)$	$96.3 \pm 1.0 (0.03, 0.07)$	$98.6 \pm 0.3 (0.02, 0.09)$	$98.9 \pm 0.2 (-, 0.00)$
Vehicle	Pure	92.7 ± 1.2	$97.9 \pm 3.3 (19)$	$94.0 \pm 2.9 (1, 0.3, 0.2)$	$97.0 \pm 4.6 (2^{-2}, 2^3)$
	Networkless	$93.2 \pm 1.9 (0.07)$	$97.9 \pm 3.3 (0.00)$	$94.8 \pm 2.8 (0.10)$	$97.2 \pm 3.7 (0.05)$
Letter	Pure	$94.9 \pm 0.4 (0.01, 0.15)$	$98.3 \pm 0.6 (0.01, 0.05)$	$96.5 \pm 1.1 (0.02, 0.21)$	$98.1 \pm 1.0 (0.01, 0.13)$
	Networkless	70.6 ± 7.7	$71.8 \pm 9.0 (1)$	$67.3 \pm 5.0 (7, 0.1, 0.3)$	$72.4 \pm 5.6 (2^{10}, 2^4)$
Vehicle	Pure	$71.5 \pm 5.7 (0.14)$	$72.7 \pm 7.1 (0.16)$	$68.8 \pm 3.2 (0.22)$	$73.3 \pm 3.9 (0.12)$
	Networkless	$79.2 \pm 5.3 (0.03, 0.32)$	$79.7 \pm 5.0 (0.35)$	$77.4 \pm 5.5 (0.02, 0.30)$	$80.1 \pm 4.3 (0.03, 0.31)$
Letter	Pure	68.1 ± 3.8	$67.6 \pm 4.1 (5)$	$69.0 \pm 4.4 (5, 0.3, 0.2)$	$84.4 \pm 3.4 (2^{10}, 2^3)$
	Networkless	$70.0 \pm 2.6 (0.19)$	$69.4 \pm 2.5 (0.18)$	$70.3 \pm 3.8 (0.13)$	$84.4 \pm 3.4 (0.00)$
Letter	Pure	74.4 ± 5.6	$96.0 \pm 7.6 (1)$	$88.9 \pm 9.9 (3, 0.2, 0.4)$	$94.8 \pm 1.7 (2^6, 2^4)$
	Networkless	$75.5 \pm 4.6 (0.14)$	$96.0 \pm 7.6 (0.00)$	$89.3 \pm 7.4 (0.09)$	$94.8 \pm 1.7 (0.00)$
Letter	Pure	$77.8 \pm 3.4 (0.04, 0.17)$	$96.0 \pm 7.6 (-, 0.00)$	$92.1 \pm 4.1 (0.04, 0.19)$	$94.8 \pm 1.7 (-, 0.00)$

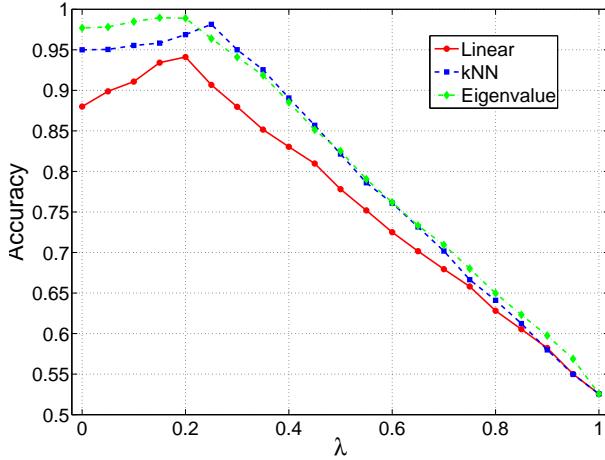


Figure 5.16: A detailed analysis of the impact of the compliance term ρ on different traditional low level techniques applied to the MNIST database. One can see that a mixture of the proposed traditional and high level techniques does give a boost in the accuracy rate in this real-world data set.

it just extends the already formed “line” pattern. As a consequence, the inclusion of the test digit into the class ‘5’ generates small variations of the component’s measures. However, if the test digit is inserted into the class ‘6’, larger variations of the component’s measures occur, since cycles are formed in the component (before insertion of the test digit, there is no cycle in the component). Figures 5.18a and 5.18b show the transient and cycle lengths, as well as the corresponding variations as a function of μ , when the test digit is inserted into the component of digit ‘5’. We see that the variations are very small, indicating the strong compliance of the test digit with the pattern formed by training digits ‘5’. On the other hand, Figs. 5.18c and 5.18d show the same information, when the test digit is put into the component of digit ‘6’. Here, we see that larger variations occur, which means that the test digit does not conform to the pattern formed by the component of digits ‘6’. As a result, the test instance is correctly classified as a digit ‘5’. The same reasoning can be applied to the digits network shown in Fig. 5.17b. In this case, the transient and cycle lengths, as well as the corresponding variations, are shown in Figs. 5.18e - 5.18h, when the test digit is inserted into the component of digit ‘5’ or ‘6’, respectively. In this situation, the test instance is classified as a digit ‘6’.

5.6 Chapter Remarks

In this work, we have proposed a novel combination of low and high level classifiers, which can be considered as a general framework, to perform supervised data classification. The low level term classifies test instances according to their physical features, while the second term measures the compliance with the pattern formation

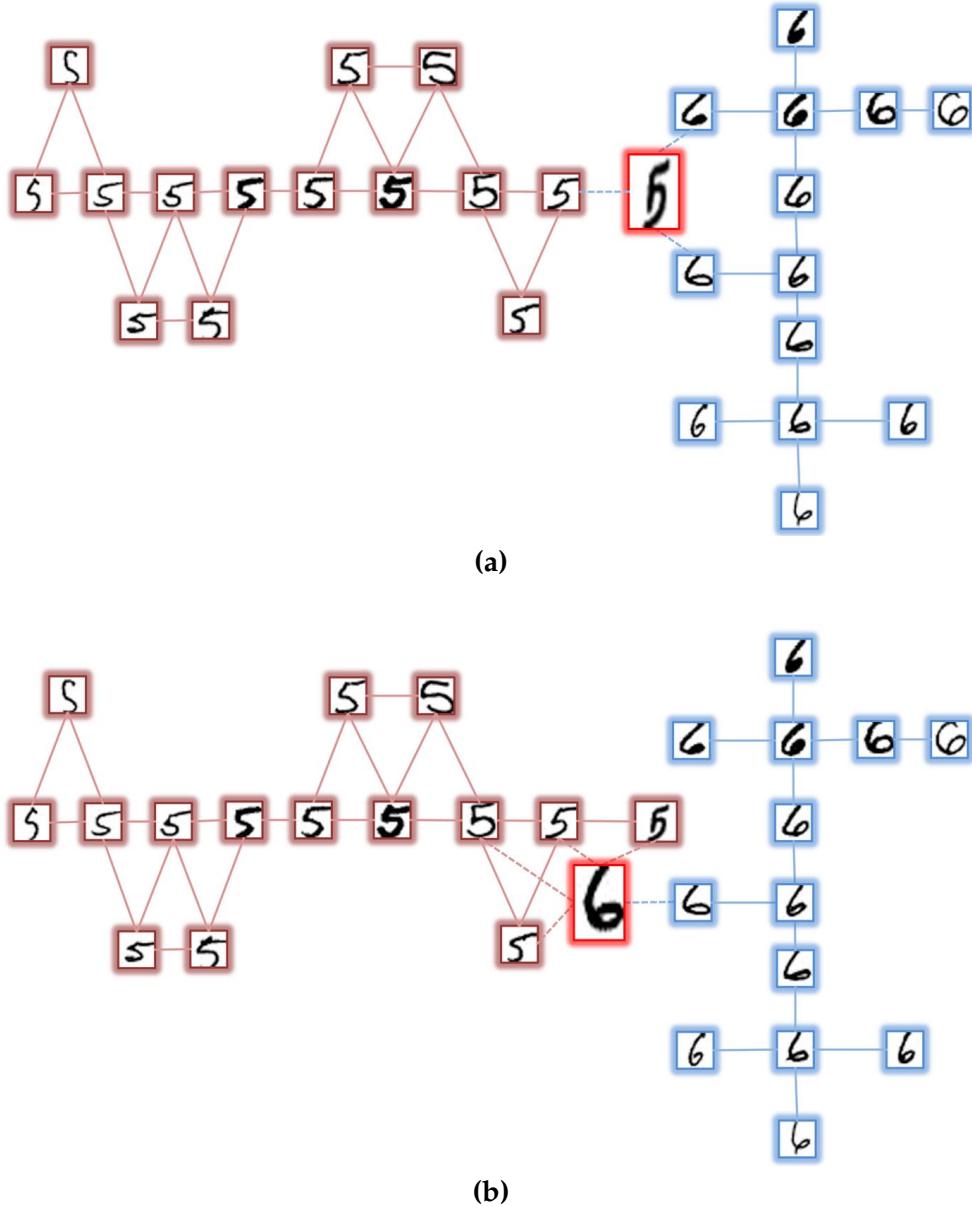


Figure 5.17: Illustration of the pattern formation impact in a subset of samples extracted from the MNIST data set. The training instances are displayed by the brown (digit 5) and blue (digit 6) colors. The test instances are indicated by a red color (bigger sizes). Insertion of a test instance whose real class is: (a) the digit 5 and (b) the digit 6.

of the training data, by means of exploiting the complex topological properties of the network. Besides the novel definition of this hybrid classifier, we have proposed two simple implementations for the high level term, both running in a networked environment. In the first one, the high level classifier is composed of three complex network measures: the average degree, clustering coefficient, and assortativity. The reason why these measures have been chosen is that they characterize the network from a local to global fashion. In the second implementation, we have utilized the complex dynamics generated by a tourist walker released in a network. Specifically, we have composed

the high level term by taking a linear weighted combination of the tourist's transient and cycle lengths for different values of the tourist's memory, up to a critical value. Likewise the previous implementation, the motivation behind taking combinations of the transient and cycle lengths for different memory values is that it can capture from local (small memory window) to global (large memory window) aspects of the network. The main contribution of this work is the proposal of a new classification scheme, which has the ability to perform high level classification according to the semantic meaning or the pattern formation of the data, as well as two simple implementations of the high level term.

Several experiments are conducted on synthetic and real-world data sets, so that we can better assess the performance of the proposed framework. A quite interesting feature of the proposed technique is that the high level term influence has to be increased in order to get correct classification as the class configuration's complexity increases. This suggests that the high level term is specially useful in complex situations of classification.

We have also explored the application of handwritten digits recognition and have found that the hybrid model can really improve the accuracy rates of traditional classifiers in certain conditions. In addition, we have confirmed that the high level term in isolation generally does not perform very well. However, when utilized together with a good low level classifier, it can really boost the performance of the latter.

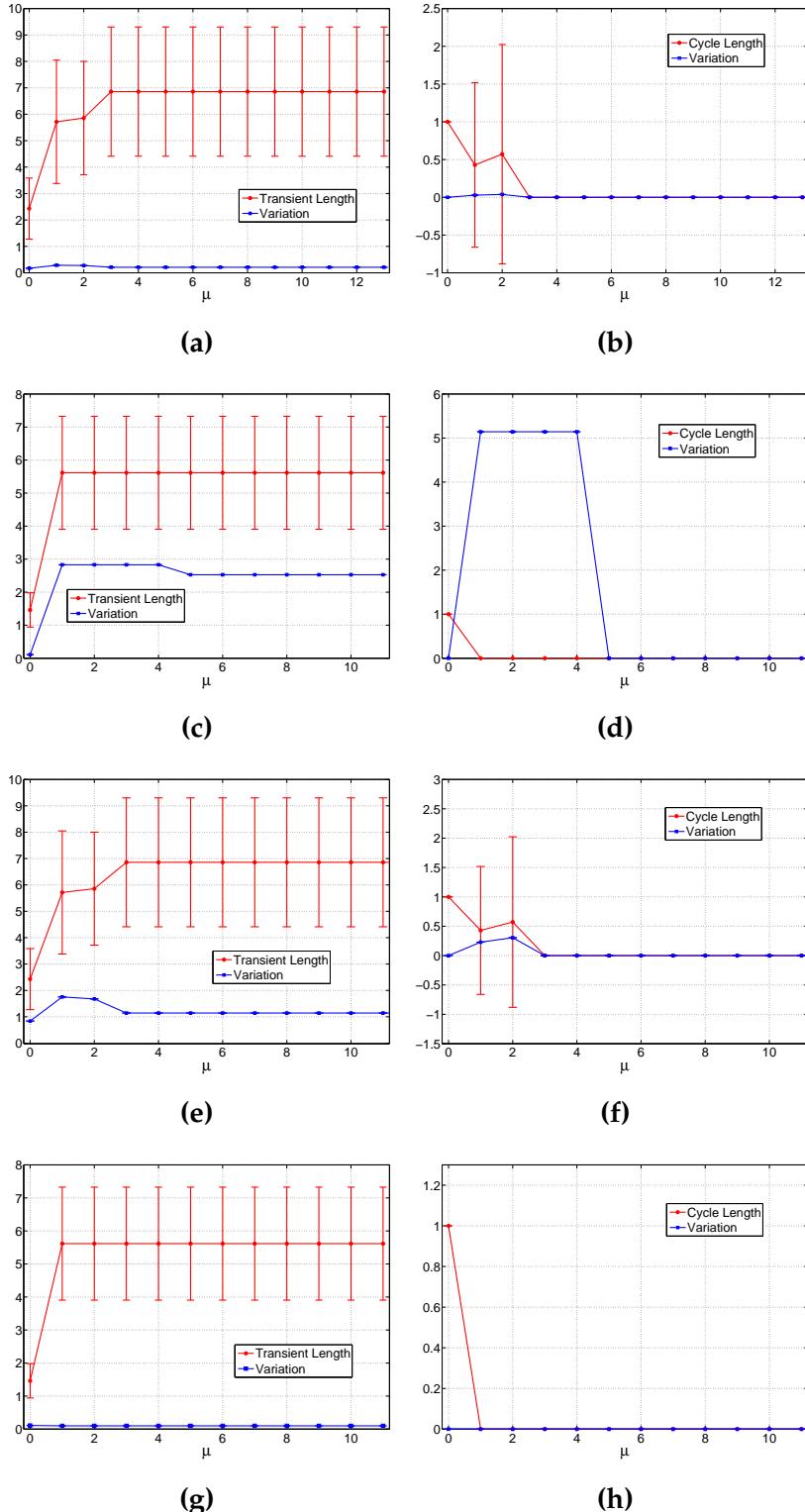


Figure 5.18: Transient and cycle lengths and the corresponding variations due to the insertion of a test instance. With regard to Fig. 5.17a: (a) and (b) transient and cycle lengths of the brown class (digit 5) and their variations by virtue of the insertion of the red instance (test instance); (c) and (d) same information for the blue class (digit 6). With regard to Fig. 5.17b: (e) and (f) transient and cycle lengths of the brown class (digit 5) and their corresponding variations due to the insertion of the red instance (test instance) with respect to the brown class; (g) and (h) same information for the blue class (digit 6).

CHAPTER
6

Conclusions

Complex networks have emerged as a unified representation of complex systems in various branches of science. In this work, we have explored how complex networks are able to enhance the learning process of machine learning techniques. The main motivation of merging these two areas of science is that the complex network representation has the ability to describe the topological structure of the original system. In the developed work, we have tackled the three main paradigms of the machine learning theory: unsupervised, semisupervised, and supervised learning. In all the approaches, the proposed techniques can satisfactorily improve the prediction power of the techniques in a consistent manner.

6.1 Concluding Remarks

In the unsupervised learning domain (Chapter 3), a novel particle competition model has been proposed. The model's description has been inspired by the prior work of [Quiles et al. \(2008\)](#), where only a procedure or sequence of steps to perform particle competition is presented, without any formal definition. The procedure of particle competition not only provides a community detection technique, but also presents a general scheme of competitive learning. Moreover, the method provides high community detection accuracy and, at the same time, low computational complexity. In its general form, the particles' movement policy is composed of a convex combination, which is weighted by the parameter λ , of two distinct components: the random and the preferential terms. The former induces the adventurous behavior of the particles, in the sense that it forces the particles to randomly choose the neighbors to visit, regardless of the domination levels registered by them. On the other hand, the latter

is responsible for the defensive behavior of the particles, in the sense that it favors the particles to revisit their already dominated vertices. The combination of them, in accordance with our computer simulation results, has revealed that the particle competition model can output decent cluster accuracy rates in both artificial and real-world data sets.

The absence of a mathematical formalization in the model of Quiles *et al.* (2008) precludes any further theoretical analysis, such as the prediction of the method's behavior for data with different natures. In contrast to that, in this project, extensive empirical and analytical investigations have been conducted in the competitive model, which has been rigorously described by a stochastic nonlinear dynamical system. An interesting feature uncovered by our mathematical analysis is that the proposed competitive model generalizes the classical model of multiple independent random walks, which is a particular form of the proposed particle competition model when $\lambda = 0$ and $\Delta = 0$. Whenever these parameters differ from zero, complex nonlinear dynamics caused by particle interactions may be encountered in each of the particles' navigation trajectories. More formally, the particles no longer can wander free in an independent way; instead, each one indirectly interacts with each other by means of the domination levels imposed on the network's vertices. This interaction may be conceived as a settle-down force, which compels the dynamical system to reach a quasi-stationary state (small fluctuations around a fixed point) in the long run. One can think of this as a strategy performed by each of the particles: to conquer a dense subgraph (community) and stay defensive on it, not letting any competitors invade its territory. As the interaction among the particles grows stronger, which can be performed by increasing λ , the lesser will be the fluctuations around this imaginary fixed point. As revealed by our convergence analysis, the proposed model does not present asymptotic stability, but instead it presents structural stability. In our opinion, this characteristic is not a disadvantage, because it is more similar to several real-world systems on account of the noises and other uncontrolled variables presented by the real environment.

With regard to the same unsupervised learning model, we have also presented an efficient method for estimating the most likely number of clusters or communities in a data set by using a proposed internal evaluator index. Such index exploits the domination level information generated by the competition process itself. As a result, if we take into account that the number of clusters is far less than the quantity of data items, the cluster number determination process does not increase the model's complexity order. Since the determination of the actual number of clusters is an important issue in data clustering, our method also presents a contribution to this topic. Following the same line, an index for detecting overlapping cluster structures is also proposed which, under some assumptions, may also not increase the model's computational complexity due to its embedded nature within the competitive process. Several com-

puter simulations have been presented, both on well-known complex networks and on benchmarked data sets.

Still in the unsupervised model, not only have we dealt with community detection tasks, but also we have delved into data clustering tasks. In this case, the raw, vector-based data is transformed into a network and, then, the proposed technique is applied. In this context, the application of handwritten digits and letters clustering has been explored, and high clustering accuracies have been obtained. Moreover, we have analyzed the composition of the clusters formed in the MNIST data set and have verified that, within a specific cluster, several variations of the same pattern can be encountered. This is expected because these variations can be captured via long chains of edges between vertices (patterns) of the same cluster. In view of this, we can confirm the robustness of the model on account of its network representation.

In the semisupervised learning domain (Chapter 4), we have extended and developed new mechanisms to adapt the particle competition model for semisupervised learning tasks. One of these new mechanisms is the introduction of cooperation among particles. In short, particles of the same class proceed in the network in a cooperative manner to propagate their labels, while particles of different classes compete with each other to determine the class borders. Another interesting feature observed in the computer simulations is that the model has a local label-spreading fashion, i.e., due to the competitive mechanism, each particle only visits a portion of vertices potentially belonging to it or its teammates. This can be roughly understood as a "divide-and-conquer" effect embedded in the competitive-cooperative scheme. In this way, many long-range redundant operations are avoided. As a result, the proposed method has a lower computational complexity order. Likewise the previous case, we have also developed a theoretical analysis of the semisupervised model and estimated probabilities for the marginal distributions have been analytically evaluated. In addition, we have studied a simple example, where we show that the theoretical analysis really approximates the empirical behavior of the model.

Furthermore, we have also investigated a new mechanism to prevent error propagation in general semisupervised learning. To our knowledge, many semisupervised learning techniques have been proposed, but the majority of them considers that the label information of the labeled subset is supposed to be completely reliable. However, in real situations, this assumption is not always true and mislabeled samples are commonly found in the data sets due to instrumental errors, corruption from noise, or even human mistakes in the labeling process. Thus, there is a necessity to develop error prevention mechanism in semisupervised learning technique. In this way, the proposed mechanism together with the analysis and numerical simulations presented in this thesis also make a clear contribution to general machine learning and especially to autonomous learning research. This topic has practical importance because the qual-

ity of the training data is a fundamental issue in semisupervised learning, since usually few labeled data are available and errors (wrong labels) may easily be propagated to a portion of or the entire data set. Specifically, the error propagation prevention mechanism can benefit the learning systems from two aspects: (i) improving the performance of the learning system, i.e., the system can learn from errors; and (ii) avoiding catastrophe of the system by limiting the spreading of wrong labels. The detection of possible wrongly labeled vertices is conducted by using a dynamical stochastic variable that is inferred from the own competitive process. Once a vertex is accused of possibly misleading the learning process (mislabeled), a corrective procedure is applied to it in order to correct its label by using the current local information. In addition, analytical and empirical analyses on how the parameters influence the model's performance have been carried out. Computer simulations revealed that the model can withstand strong noises from the environment by means of the interactive competitive process performed by the particles.

In the supervised learning domain (Chapter 5), we have proposed a novel hybrid framework that is composed of a low and high level classifiers. The former is represented by techniques that use physical features of the input data, while the latter exploits the class pattern formation in the learning process. Usually, the data items are not isolated points in the attribute space, but instead tend to form certain patterns. The human (animal) brain performs both low and high orders of learning and it has facility in identifying patterns according to the semantic meaning of the input data. However, this kind of task, in general, is still cumbersome to be assessed by computers. This is exactly what the second term, the high level classifier, tries to capture, i.e., it not only considers physical attributes of the input data, but also the pattern formation in the inference process. Both classifiers are combined by utilizing a convex combination, weighted by what we call the compliance term. The literature provides a myriad of low level classifiers, such as the linear discriminator, neural networks, SVM, decision trees, among many others; however, little can be found that deal with high order of learning. Having this gap in mind, we have also proposed two simple network-based high level classification methods, where each of the classifiers attempts to exploit the complex topological properties of the underlying network constructed from the input data. In the first classifier, we have used the well-known assortativity, clustering coefficient, and average degree for conducting the high order of learning. We have shown that the combination of these three measures can capture from local to global characteristics of the network. In the second, we have applied a weighted combination of tourist walks with different memory lengths. For this end, we have used the variations of the transient and cycle lengths of tourist walks for varying values of the tourist's memory. We have confirmed that it is able to capture complex patterns of the network and is able to provide reasonable accuracy rates when utilized in conjunction with a

good low level classifier.

Still in the supervised learning, we have applied the hybrid framework to the application of handwritten digits and letters recognition using a real-world data set. We have seen that improved results can be obtained if one utilizes a mixture of the low and high level classifiers, rather than one in isolation. A quite interesting feature observed in these simulations is that, in the proposed technique, the high level term influence has to be increased in order to get correct classification as the class configuration's complexity increases. However, in simple situations where the cluster assumption holds, a low level classifier may perform better. This suggests that the high level term is specially useful in complex situations of classification.

6.2 Publications during the Doctorate Period

During the doctorate period, several research results have been obtained and a series of papers have been published. The main results have been published in the *IEEE Transactions on Neural Networks and Learning Systems* ([Silva and Zhao, 2012a,b,d](#)), which is one of the leading scientific journals with impact factor 2.952 currently. The complete list is shown as follows:

- Eight publications in international journals (see ([Cupertino et al., 2012a](#); [Silva and Amancio, 2012](#); [Silva and Zhao, 2012a,b,c,d,f](#); [Silva et al., 2012c](#))):
 1. **Thiago C. Silva and Liang Zhao (2012)**, "Stochastic Competitive Learning in Complex Networks," *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):385-398⁸.
 2. **Thiago C. Silva and Liang Zhao (2012)**, "High Level Classification in Complex Networks," *IEEE Transactions on Neural Networks and Learning Systems*, 23(6):954-970.
 3. **Thiago C. Silva and Liang Zhao (2012)**, "Network-Based Stochastic Semisupervised Learning," *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):451-466.
 4. **Thiago C. Silva and Liang Zhao (2012)**, "Detecting and Preventing Error Propagation via Competitive Learning," *Neural Networks*, DOI: 10.1016/j.neunet.2012.11.001.
 5. **Thiago C. Silva and Liang Zhao (2012)**, "Semisupervised Learning Guided by the Modularity Measure in Complex Networks," *Neurocomputing*, 78(1):30-37.

⁸This work has received the award of CIS Publication Spotlight of the IEEE Transactions on Neural Networks and Learning Systems in the IEEE Computational Intelligence Magazine ([Liu et al., 2012](#)).

6. **Thiago C. Silva, Liang Zhao and Thiago H. Cupertino (2012)**, "Handwritten Data Clustering using Agents Competition in Networks," *Journal of Mathematical Imaging and Vision*, DOI: 10.1007/s10851-012-0353-z.
 7. **Thiago C. Silva and Diego R. Amancio (2012)**, "Word sense disambiguation via high order of learning in complex networks," *Europhysics Letters*, 98(5):58001.
 8. **Thiago H. Cupertino, Thiago C. Silva, and Liang Zhao (2012)**, "Classification of Multiple Observation Sets via Network Modularity," *Neural Computing and Applications*, DOI: 10.1007/s00521-012-1115-y.
- Eleven publications in international and national conferences (c.f. ([Araújo *et al.*, 2010](#); [Cupertino *et al.*, 2012b](#); [Silva and Zhao, 2012e,g, 2011a,b,c](#); [Silva *et al.*, 2011a,b, 2012a,b](#))).

6.3 Scientific Contributions

The product of this project has rendered several contributions to the machine learning community, in the three paradigms of learning. These results have been obtained by utilizing the fact that complex networks are able to describe the topological structure of the data, permitting one to detect clusters and classes in different sizes, shapes and densities in a consistent form, and, therefore, improve the learning process of a machine learning task.

In relation to the unsupervised learning area, we can enunciate the following contributions:

1. A particle competition model has been designed, which relies on a stochastic nonlinear dynamical system to perform the learning process;
2. An analytical analysis of the model has been performed. Since the model of interacting particles corresponds to many natural and artificial systems, the study of this topic stands as an important task. Due to the scarcity of theories for such models, this work is an important step to understand these kinds of systems. In addition, we have shown that the proposed system is a generalization of the case of multiple independent random walks;
3. A convergence analysis of the model has been conducted. Interestingly, we have seen that the model does not present asymptotic stability, but instead it presents structural stability. In our opinion, this characteristic is not a disadvantage, because it is more similar to several real-world systems on account of the noises and other uncontrolled variables presented by the real environment;

4. The proposed technique may run in linear time complexity when applied to sparse networks, which is more efficient in relation to what the literature offers until now;
5. Several computer simulations have been performed, both on synthetic and real-world data sets. The application of the proposed technique on known network-based benchmarks has confirmed the effectiveness of the technique;
6. Parameter selection guidelines have been studied, which enable one to better tune the algorithms for his/her needs;
7. An embedded evaluator index for estimating the most likely number of clusters or communities in a data set has been proposed. We have shown that, under some circumstances, its employment may not increase the model's time complexity;
8. An embedded evaluator index for detecting the overlapping characteristics of the clusters and vertices has been designed. Likewise the previous item, this may also not increase the model's time complexity;
9. We have shown that the particle competition model may capture various distortions of handwritten digits and letters in the same cluster. This is made possible by virtue of the long chains of edges linking vertices (patterns) in the same cluster or community.

In what concerns the semisupervised learning domain, the following contributions have been made:

1. A network-based semisupervised learning technique has been proposed, which is an extension of the previous unsupervised particle competition model. A cooperative scheme has been introduced to enable teams of particles to cooperate with each other;
2. An analytical analysis of the model has also been performed, permitting the understanding of the complex dynamics of the competitive model. An example linking the theoretical and the empirical approaches confirms our results;
3. An interesting feature is that the model has a local label-spreading fashion, i.e., due to the competitive mechanism, each particle only visits a portion of vertices potentially belonging to the current particle or its teammates. This can be roughly understood as a "divide-and-conquer" effect embedded in the competitive-cooperative scheme. In this way, many long-range redundant operations are avoided. As a result, the proposed method has a lower computational complexity order;

4. An efficient mechanism for detecting and preventing error propagation has been presented. Since this topic is of little investigation in the literature and the propagation of mislabeled vertices can provoke a disaster in the learning process, it turns out to be an important study;
5. Guidelines of parameter selection have been offered for the new stochastic dynamical variables introduced in the error propagation prevention scheme. We have made analytical and empirical analyses of the behavior of the technique for different parameter values;
6. We have shown that the error propagation prevention scheme can withstand environments with heavy noises. Moreover, we have seen that the competitive process among the particles is the principal actor of vanishing the mislabeled vertices, when these are presented as the minority of the labeled instances;
7. Several computer simulations have been performed, both on artificial and real-world data sets, including simulations performed on network-based semisupervised benchmarks.

Finally, with regard to the supervised learning paradigm, the following contributions can be listed:

1. A hybrid framework has been proposed to link both the low and high levels of learning. The low level classifier takes into account the classical assumptions of the classifiers, such as the cluster or smoothness assumptions. On the other hand, the high level term exploits the complex properties of a network constructed from the input data to infer its decision by measuring how test instances comply with the training instances;
2. Since few studies have been done to develop high level classifiers, we have proposed two network-based techniques to realize this objective. In the first technique, we have used three network measures, namely, assortativity, clustering coefficient, and average degree, to perform the high order of learning. We have shown that the combination of these three measures can capture from local to global characteristics of the network. In the second technique, we have applied a weighted combination of tourist walks with different memory lengths. For this end, we have used the variations of the transient and cycle lengths of tourist walks for varying values of the tourist's memory. In this way, it is able to capture complex patterns of the network and is able to provide reasonable accuracy rates when utilized in conjunction with a good low level classification technique;

3. Improvements of the low level classification can be achieved by the combination of the two levels of learning. Furthermore, as the class configuration's complexity increases, such as the mixture among different classes, a larger portion of the high level term is required to correctly determine the pattern formation. Particularly and also intuitively, the high level term is not required in the case where all classes are completely separated and each class is well-posed;
4. A real-world application has been explored, where the proposed hybrid framework is able to identify variations of handwritten digits images and, consequently, improve the pattern recognition rate;
5. Even though the high level techniques are simple, the work that we have shown here makes clear that the hidden organization of patterns within data sets can be effectively used to improve the accuracy rate of classification tasks.

6.4 Future Works

With regard to the particle competition model, several extensions could be made, including:

1. One could use different quantities of particles to enable hierarchical clustering. For example, as more particles are introduced into the network, more fine-grained the clusters will be. As this quantity is decremented, the clusters are joined together to compose larger clusters (agglomerative approach). Similarly, one could start out by using small quantities of particles and then increase this value (divisive approach);
2. Each particle may only perform a movement, i.e., travel through a link, at each time step (iteration). We believe that, using a local strategy, the particles could perform more movements at each time step, depending on the circumstances of the region which it is visiting. This could help the model to get more robust cluster accuracy rates (unsupervised learning) or classification accuracy rates (semisupervised learning);
3. The particle competition model could be extended to the supervised learning domain. In this case, one could classify the test instances by using the information generated in the training phase, such as the domination level of the test instance's neighboring vertices;
4. The particles perform movements in accordance with a stochastic combination of random and preferential terms, weighted by a fixed parameter λ . One could

think of using varying λ , i.e., $\lambda(t)$. A simple idea would be to let λ small in the start (so the random walks would initially prevail), then start to increase its value as time progresses (so the preferential walks would predominate in the long run). This strategy would eliminate the quasi-stationary state of model, turning it into a fixed point, since, in the long-run, the particles would no longer have the adventurous behavior, which is performed by the random term;

5. The particle competition model could be applied to data stream problems, where the networks evolve as time progresses. One could use summarized information generated from the competition process to efficiently cluster new incoming instances;
6. Another interesting idea is to use an adaptive quantity of particles in the network. The model starts out by putting several particles into the network. By using a specific measure, one could eliminate particles that are not being effective enough. For example, if a particle is getting exhausted very often, this is a signal that a fierce competition in the network is taking place. With this respect, this is a good indication that there are more particles in the network than communities. Ideally, this measure would settle down when the number of the particles is somewhat equal to the number of communities. We believe this would bring complex and interesting behavior in the stochastic dynamical processes.

In relation to the hybrid classification technique, the following extensions could be investigated:

1. The high level classification could be extended by the addition of new network measures, such as degree entropy, component spectrum, average edge reciprocity, matching indices, among many others (see ([Costa et al., 2007](#)) for details about these measures);
2. As we have described, the network formation technique is critical to make the proposed method working. In this aspect, new network formation techniques could be developed, possibly by using adaptive ϵ and k in the training and classification phases;
3. A major drawback of the proposed model is its time complexity. For each test instance, we need to recalculate the variations of its insertion in all the class components. A good idea would be to use sample strategies when classifying test instances. This would enable the treatment of large-scale networks;
4. One could think of using complex network synchronization theories when designing the high level classifier. With this respect, each vertex could be considered

as an oscillator. When a test instance is inserted, we could quantify its compliance by checking with which class it synchronizes;

5. New real-world applications could be explored.

Bibliography

Abe and Inoue(2002) Shigeo Abe and Takuya Inoue. Fuzzy support vector machines for multiclass problems. In *European Symposium on Artificial Neural Networks, 2002*, pages 113–118. Cited in page(s) [231](#)

Albert and Barabási(2002) Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97. Cited in page(s) [3](#)

Albert et al.(1999) Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world wide web. *Nature*, 401:130–131. Cited in page(s) [19](#)

Albert et al.(2004) Réka Albert, Istvan Albert, and Gary L. Nakarado. Structural vulnerability of the north american power grid. *Physical Review E*, 69:025103. Cited in page(s) [19, 20](#)

Allinson et al.(2001) Nigel Allinson, Hujun Yin, Lesley Allinson, and Jon Slack. *Advances in Self-Organising Maps*. Springer. Cited in page(s) [2](#)

Alpaydin(2004) Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, second edition. Cited in page(s) [36, 38, 39, 55](#)

Amini and Gallinari(2003) Massih-Reza Amini and Patrick Gallinari. Semi-supervised learning with explicit misclassification modeling. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 555–560, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Cited in page(s) [11, 175](#)

Amini and Gallinari(2005) Massih-Reza Amini and Patrick Gallinari. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8(4):385–413. ISSN 0219-1377. Cited in page(s) [11, 175](#)

Amorim et al.(2007) Dinani G. Amorim, Manuel F. Delgado, and Senén B. Ameneiro. Polytope ARTMAP: Pattern classification without vigilance based on general geometry categories. *IEEE Transactions on Neural Networks*, 18(5):1306–1325. Cited in page(s) [2](#)

Araújo et al.(2010) Bilzâ M. Araújo, Francisco A. Rodrigues, Thiago C. Silva, and Liang Zhao. Identifying abnormal nodes in protein-protein interaction networks. In *XI Brazilian Symposium on Artificial Neural Network (SBRN)*, pages 97–102. Cited in page(s) [258](#)

Arenas et al.(2006) Alex Arenas, Albert D. Guilera, and Conrad J. Pérez Vicente. Synchronization reveals topological scales in complex networks. *Physical Review Letters*, 96(11):114102. Cited in page(s) [4](#), [26](#), [28](#), [40](#)

Arenas et al.(2008) Alex Arenas, Alberto Fernández, and Sergio Gómez. Limited resolution in complex network community detection with potts model approach. *New Journal of Physics*, 10:053039. Cited in page(s) [43](#)

Athinarayanan et al.(2002) Ragu Athinarayanan, Mohammad R. Sayeh, and Dale A. Wood. Adaptive competitive self-organizing associative memory. *IEEE Transactions on Systems, Man and Cybernetics , Part A*, 32(4):461–471. Cited in page(s) [2](#)

Bacciu and Starita(2008) Davide Bacciu and Antonina Starita. Competitive repetition suppression (CoRe) clustering: A biologically inspired learning model with application to robust clustering. *IEEE Transactions on Neural Networks*, 19(11):1922–1940. Cited in page(s) [2](#)

Barabási and Albert(1999) Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science - New York*, 286(5439):509–512. Cited in page(s) [3](#), [19](#), [24](#)

Barabási et al.(2002) Albert-László Barabási, Hawoong Jeong, Zoltan Neda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4): 590 – 614. Cited in page(s) [47](#)

Barrat et al.(2008) Alain Barrat, Marc Barthélémy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press. Cited in page(s) [3](#)

Belkin and Niyogi(2003) Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396. Cited in page(s) [10](#), [171](#)

Belkin et al.(2004) Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semisupervised learning on large graphs. In *Conference on Learning Theory*, pages 624–638. Springer. Cited in page(s) [10](#), [52](#), [57](#), [171](#)

Belkin et al.(2005) Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. On manifold regularization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, pages 17–24, New Jersey. Society for Artificial Intelligence and Statistics. Cited in page(s) [52](#), [57](#)

Belkin et al.(2006) Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434. Cited in page(s) [47](#), [52](#), [57](#)

Berkhin(2002) Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software. Cited in page(s) [39](#)

Bertini Junior et al.(2011) João R. Bertini Junior, Liang Zhao, Robson Motta, and Alneu A. Lopes. A nonparametric classification method based on K-Associated graphs. *Information Sciences*, 181:5435–5456. Cited in page(s) [47](#), [49](#), [50](#)

- Bezdek(1981)** James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, first edition. Cited in page(s) [81](#), [123](#)
- Binaghi et al.(2003)** Elisabetta Binaghi, Ignazio Gallo, and Monica Pepe. A cognitive pyramid for contextual classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 41(12):2906–2922. Cited in page(s) [202](#)
- Bishop(2007)** Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, second edition. Cited in page(s) [1](#), [35](#), [36](#), [38](#), [39](#), [44](#), [133](#), [136](#)
- Blum and Chawla(2001)** Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 19–26, San Francisco. Morgan Kaufmann. Cited in page(s) [57](#)
- Blum and Mitchell(1998)** Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100. Cited in page(s) [201](#)
- Boccaletti et al.(2007)** Stefano Boccaletti, Mikhail V. Ivanchenko, Vito Latora, Alessandro Pluchino, and Andrea Rapisarda. Detecting complex network modularity by dynamical clustering. *Physical Review E*, 75(4):045102. Cited in page(s) [9](#)
- Bollobas(1998)** Bela Bollobas. *Modern Graph Theory*. Springer, corrected edition. Cited in page(s) [16](#)
- Boriah et al.(2008)** Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *SIAM Data Mining Conference*, pages 243–254. Cited in page(s) [230](#)
- Bornholdt and Schuster(2003)** Stefan Bornholdt and Heinz G. Schuster. *Handbook of Graphs and Networks: From the Genome to the Internet*. Wiley-VCH. Cited in page(s) [19](#)
- Burges and Platt(2006)** Christopher J. C. Burges and John C. Platt. *Semi-supervised Learning. ch. Semi-Supervised Learning with Conditional Harmonic Mixing*, pages 251–273. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA. Cited in page(s) [171](#)
- Callut et al.(2008)** Jérôme Callut, Kevin Françoise, Marco Saerens, and Pierre Dupont. Semi-supervised classification from discriminative random walks. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, LNAI 5211*:162–177. Cited in page(s) [57](#), [192](#), [193](#)
- Capocci et al.(2004)** Andrea Capocci, Vito D. P. Servedio, Guido Caldarelli, and Francesca Colaiori. Communities detection in large networks. In *Workshop on Algorithms and Models for the Web Graph (WAW)*, pages 181–188. Cited in page(s) [80](#)
- Carpenter and Grossberg(1987)** Gail A. Carpenter and Stephen Grossberg. Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930. Cited in page(s) [2](#)

- Cha(2007)** Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1:300–307. Cited in page(s) [93](#), [95](#)
- Chapelle et al.(2003)** Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *The Conference on Neural Information Processing Systems (NIPS)*, volume 15, pages 585–592, Cambridge, MA, USA. MIT Press. Cited in page(s) [171](#)
- Chapelle et al.(2006)** Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA. Cited in page(s) [2](#), [12](#), [35](#), [47](#), [52](#), [53](#), [54](#), [55](#), [56](#), [163](#), [165](#), [170](#), [171](#), [175](#), [194](#), [198](#), [202](#)
- Chatfield(2009)** Mark Chatfield. The skillings-mack test (friedman test when there are missing data). *Stata Journal*, 9(2):299–305(7). Cited in page(s) [172](#)
- Chau and Basu(2011)** Chi-Kin Chau and Prithwish Basu. Analysis of latency of stateless opportunistic forwarding in intermittently connected networks. *IEEE/ACM Transactions on Networking*, 19(4):1111–1124. Cited in page(s) [9](#), [10](#)
- Chen et al.(2009)** Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012. Cited in page(s) [47](#)
- Chen et al.(2005)** Ming Chen, Ali A. Ghorbani, and Virendrakumar C. Bhavsar. Incremental communication for adaptive resonance theory networks. *IEEE Transactions on Neural Networks*, 16(1):132–144. Cited in page(s) [2](#)
- Chung(1997)** Fan R. K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society. Cited in page(s) [16](#), [38](#)
- Çinlar(1975)** Erhan Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, Englewood Cliffs, N. J. Cited in page(s) [66](#), [69](#), [107](#), [108](#)
- Cinque et al.(2004)** Luigi Cinque, Gian L. Foresti, and Luca Lombardi. A clustering fuzzy approach for image segmentation. *Pattern Recognition*, 37:1797–1807. Cited in page(s) [38](#)
- Clauset et al.(2004)** Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111+. Cited in page(s) [32](#), [38](#), [42](#), [43](#), [59](#), [80](#), [81](#), [123](#)
- Cooper et al.(2009)** Colin Cooper, Alan Frieze, and Tomasz Radzik. Multiple random walks in random regular graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1738–1761. Cited in page(s) [10](#)
- Corduneanu and Jaakkola(2006)** Adrian Corduneanu and Tommi Jaakkola. *Semi-supervised Learning. ch. Data-Dependent Regularization*, pages 163–190. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA. Cited in page(s) [171](#)
- Cortes and Vapnik(1995)** Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, pages 273–297. Cited in page(s) [55](#)

- Costa et al.(2007)** Luciano da F. Costa, Francisco A. Rodrigues, Gonzalo Travieso, and Paulino R. Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242. Cited in page(s) [22](#), [262](#)
- Cover and Hart(1967)** Thomas M. Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21– 27. Cited in page(s) [46](#)
- Cupertino et al.(2012a)** Thiago H. Cupertino, Thiago C. Silva, and Liang Zhao. Classification of multiple observation sets via network modularity. *Neural Computing and Applications*. doi: 10.1007/s00521-012-1115-y. Cited in page(s) [257](#)
- Cupertino et al.(2012b)** Thiago H. Cupertino, Thiago C. Silva, and Liang Zhao. Multiple images set classification via network modularity. In *Workshop of Theses and Dissertations (WTD) in SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images)*, pages 124–129. Cited in page(s) [258](#)
- Danon et al.(2005)** Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008. Cited in page(s) [4](#), [26](#), [29](#), [40](#), [79](#), [114](#), [117](#), [118](#)
- Danon et al.(2007)** Leon Danon, Jordi Duch, Alex Arenas, and Díaz-Guilera. Community structure identification in large scale structure and dynamics of complex networks: From information technology to finance and natural science. *World Scientific Publishing Co.*, pages 93–113. Cited in page(s) [9](#)
- Dara et al.(2002)** Rozita Dara, Stefan C. Kremer, and Deborah A. Stacey. Clustering unlabeled data with SOMs improves classification of labeled real-world data. In *Proceedings of the World Congress on Computational Intelligence (WCCI)*, pages 2237–2242. Cited in page(s) [55](#)
- Deboeck and Kohonen(2010)** Guido J. Deboeck and Tuevo K. Kohonen. *Visual Explorations in Finance: with Self-Organizing Maps*. Springer, first edition. Cited in page(s) [2](#)
- Delalleau et al.(2005)** Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. *Efficient Non-Parametric Function Induction in Semi-Supervised Learning*, pages 96–103. Society for Artificial Intelligence and Statistics. Cited in page(s) [171](#)
- Demiriz et al.(1999)** Ayhan Demiriz, Kristin P. Bennett, and Mark J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks in Engineering (ANNIE-99)*, pages 809–814. ASME Press. Cited in page(s) [55](#)
- Dempster et al.(1977)** Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38. Cited in page(s) [81](#), [123](#)
- Demšar(2006)** Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30. Cited in page(s) [124](#), [137](#), [138](#), [165](#), [166](#), [172](#), [194](#)
- Diestel(2006)** Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer. Cited in page(s) [16](#)

- do Rêgo *et al.*(2010)** Renata L. M. E. do Rêgo, Aluizio F. R. Araújo, and Fernando B. L. Neto. Growing self-reconstruction maps. *IEEE Transactions on Neural Networks*, 21(2):211–223. Cited in page(s) [2](#)
- Dorogovtsev and Mendes(2003)** Sergey N. Dorogovtsev and José F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW (Physics)*. Oxford University Press, USA, first edition. Cited in page(s) [20](#), [47](#)
- Duch and Arenas(2005)** Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104. Cited in page(s) [80](#)
- Duda *et al.*(2000)** Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, second edition. Cited in page(s) [1](#), [35](#)
- Duda *et al.*(2001)** Richard O. Duda, Peter E. Hart, and David G. Stork. Unsupervised learning and clustering. In *Pattern Classification*. Wiley-Interscience, second edition. Cited in page(s) [39](#), [133](#)
- Duin(2000)** Robert P. W. Duin. PRTools - a MATLAB toolbox for pattern recognition. *Proceedings of SPIE*, page 1331. Cited in page(s) [121](#), [161](#), [227](#)
- Erdős and Rényi(1959)** Paul Erdős and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297. Cited in page(s) [3](#), [19](#), [20](#)
- Evans and Lambiotte(2009)** Tim S. Evans and Renaud Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):016105. Cited in page(s) [10](#), [28](#)
- Faloutsos *et al.*(2004)** Christos Faloutsos, Kevin S. Mccurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 118–127. ACM Press. Cited in page(s) [48](#)
- Faloutsos *et al.*(1999)** Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM 99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, volume 29, pages 251–262, New York, NY, USA. ACM. Cited in page(s) [19](#)
- Feigenbaum *et al.*(2007)** Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. The semantic web in action. *Scientific American*, 297(6):90–97. Cited in page(s) [201](#)
- Fortunato(2010)** Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174. Cited in page(s) [3](#), [4](#), [10](#), [26](#), [27](#), [28](#), [40](#), [79](#), [119](#), [127](#)
- Fortunato and Barthélémy(2007)** Santo Fortunato and Marc Barthélémy. Resolution limit in community detection. In *Proceedings of the National Academy of Science of the USA*, volume 104, page 36–41. Cited in page(s) [43](#)
- Fortunato *et al.*(2004)** Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical Review E*, 70(5):056104. Cited in page(s) [4](#), [28](#), [40](#)

- Frank and Asuncion(2010)** Andrew Frank and Arthur Asuncion. UCI machine learning repository, 2010. Cited in page(s) [93](#), [123](#), [163](#), [193](#), [230](#)
- Freeman(1977)** Linton C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41. Cited in page(s) [31](#)
- Freeman(2004)** Linton C. Freeman, editor. *The Development of Social Network Analysis*. Adaptive computation and machine learning. Empirical Press, Vancouver. Cited in page(s) [31](#)
- Frey and Dueck(2007)** Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976. Cited in page(s) [27](#)
- Fu and Wang(2003)** Xiuju Fu and Lipo Wang. Data dimensionality reduction with application to simplifying rbf network structure and improving classification performance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):399–409. Cited in page(s) [127](#)
- Gallagher et al.(2008)** Brian Gallagher, Hanghang Tong, Tina Eliassi-rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Knowledge Discovery and Data Mining*, pages 256–264. Cited in page(s) [202](#)
- Gan(2007)** Guojun Gan. *Data clustering: theory, algorithms, and applications*, volume 20. Society for Industrial and Applied Mathematics. Cited in page(s) [36](#), [38](#), [39](#)
- Gärtner(2008)** Thomas Gärtner. *Kernels for Structured Data*, volume 72. World Scientific Publishing Co., first edition. Cited in page(s) [55](#)
- Girvan and Newman(2002)** Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826. Cited in page(s) [25](#), [28](#), [35](#), [80](#), [120](#), [127](#), [130](#), [188](#)
- Godsil and Royle(2001)** Christopher D. Godsil and Gordon Royle. *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer. Cited in page(s) [16](#)
- Goldhirsch et al.(1987)** Isaac Goldhirsch, Steven A. Orszag, and Bhalara K. Maulik. An efficient method for computing leading eigenvalues and eigenvectors of large asymmetric matrices. *Journal of Scientific Computing*, 2:33–58. Cited in page(s) [134](#)
- Gori et al.(2005)** Marco Gori, Marco Maggini, and Lorenzo Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):167–256. Cited in page(s) [2](#)
- Govindan and Shivaprasad(1990)** Vishnupriya K. Govindan and Adamane P. Shivaprasad. Character recognition: A review. *Pattern Recognition*, 23:671–683. Cited in page(s) [133](#)
- Grady(2006)** Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783. ISSN 0162-8828. Cited in page(s) [2](#), [57](#)

Gross and Yellen(1999) Jonathan Gross and Jay Yellen. *Graph theory and its applications*. CRC Press, Inc., Boca Raton, FL, USA. Cited in page(s) [16](#)

Grossberg(1987) Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63. Cited in page(s) [2](#)

Gulbahce and Lehmann(2008) Natali Gulbahce and Sune Lehmann. The art of community detection. *BioEssays*, 30(10):934–938. Cited in page(s) [26](#)

Hartono and Hashimoto(2007) Pitoyo Hartono and Shuji Hashimoto. Learning from imperfect data. *Applied Soft Computing*, 7(1):353–363. ISSN 1568-4946. Cited in page(s) [11](#), [175](#)

Hasan et al.(2006) Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Proceedings of SDM 06 workshop on Link Analysis, Counterterrorism and Security*. Cited in page(s) [47](#), [48](#)

Hastie et al.(2009) Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition edition. Cited in page(s) [38](#), [231](#)

Haykin(2008) Simon S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition. Cited in page(s) [45](#), [201](#)

Hofman and Wiggins(2008) Jake M. Hofman and Chris H. Wiggins. Bayesian approach to network modularity. *Physical Review Letters*, 100(25):258701+. Cited in page(s) [27](#)

Hsu and Lin(2002) Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425. Cited in page(s) [218](#), [232](#)

Hull(1994) Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550–554. Cited in page(s) [134](#)

Husek et al.(2006) Dušan Husek, Jaroslav Pokorný, Hana Rezanková, and Václav Snášel. Data clustering: From documents to the web. In *Web Data Management Practices: Emerging Techniques and Technologies*, pages 1–33. IGI Global. Cited in page(s) [39](#)

Jain(2010) Anil K. Jain. Data clustering: 50 years beyond K-Means. *Pattern Recognition Letters*, 31:651–666. Cited in page(s) [38](#), [39](#), [47](#)

Jain et al.(1999) Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Survey*, 31(3):264–323. Cited in page(s) [36](#), [38](#), [39](#)

Jain et al.(2010) Lakhmi C. Jain, Beatrice Lazzerini, and Halici Ugur. *Innovations in ART Neural Networks (Studies in Fuzziness and Soft Computing)*. Physica-Verlag, Heidelberg. Cited in page(s) [2](#)

- Jensen *et al.*(2004)** David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598. Cited in page(s) [48](#)
- Jeong *et al.*(2000)** Hawoong Jeong, Bálint Tombor, Réka Albert, Zoltán N. Oltvai, and Albert-László Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654. Cited in page(s) [20](#)
- Jiang and Wang(2000)** Danchi Jiang and Jun Wang. On-line learning of dynamical systems in the presence of model mismatch and disturbances. *IEEE Trans. Neural Networks*, 11(6):1272–1283. Cited in page(s) [2](#)
- Jin and Pawson(2012)** Jing Jin and Tony Pawson. Modular evolution of phosphorylation-based signalling systems. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 367(1602):2540–55. Cited in page(s) [26](#)
- Joachims(2003)** Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of International Conference on Machine Learning*, pages 290–297. AAAI Press. Cited in page(s) [57](#), [171](#)
- Jolliffe(2002)** Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, first edition. Cited in page(s) [38](#), [134](#)
- Kang *et al.*(2011)** U Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. PE-GASUS: mining peta-scale graphs. *Journal Knowledge and Information Systems*, 27(2):303–325. Cited in page(s) [10](#)
- Karypis(2003)** George Karypis. *Cluto: A Clustering Toolkit*. University of Minnesota, Department of Computer Science, 2003. Cited in page(s) [123](#)
- Karypis *et al.*(1999)** George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75. Cited in page(s) [3](#), [35](#), [55](#), [121](#), [202](#)
- Kashef and Kamel(2009)** Rasha Kashef and Mohamed S. Kamel. Enhanced bisecting K-Means clustering using intermediate cooperation. *Pattern Recognition*, 42(11):2557–2569. Cited in page(s) [38](#)
- Kaylani *et al.*(2010)** Assem Kaylani, Michael Georgopoulos, Mansooreh Mollaghasemi, Georgios C. Anagnostopoulos, Christopher Sentelle, and Mingyu Zhong. An adaptive multiobjective approach to evolving ART architectures. *IEEE Transactions on Neural Networks*, 21(4):529–550. Cited in page(s) [2](#)
- Kennedy and Eberhart(1995)** James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. Cited in page(s) [230](#)
- Khan(2004)** Shehroz S. Khan. Cluster center initialization algorithm for K-Means clustering. *Pattern Recognition Letters*, 25(11):1293–1302. Cited in page(s) [81](#), [121](#), [123](#)

Kinouchi *et al.*(2002) Osame Kinouchi, Alexandre S. Martinez, Gilson F. Lima, Gisele M. Lourenço, and Sebastian Risau-Gusman. Deterministic walks in random networks: an application to thesaurus graphs. *Physica A*, 315:665–676. Cited in page(s) [34](#), [213](#)

Kiss *et al.*(1973) George R. Kiss, Christine Armstrong, Robert Milroy, and James R. I. Piper. An associative thesaurus of English and its computer analysis. In *The computer and literary studies*. University Press. Cited in page(s) [27](#)

Knuth(1993) Donald E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. ACM, New York, NY, USA. Cited in page(s) [129](#)

Kohonen(1990) Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480. Cited in page(s) [2](#)

Korb and Nicholson(2010) Kevin B. Korb and Ann E. Nicholson. *Bayesian Artificial Intelligence*. Chapman and Hall, 2nd edition. Cited in page(s) [46](#)

Kosko(1991) Bart Kosko. Stochastic competitive learning. *IEEE Transactions on Neural Networks*, 2(5):522–529. Cited in page(s) [2](#)

Kumpula *et al.*(2007) Jussi M. Kumpula, Jari Saramäki, Kimmo Kaski, and Janos Kertész. Limited resolution in complex network community detection with Potts model approach. *The European Physical Journal B*, 56:41–45. Cited in page(s) [43](#)

Lancichinetti and Fortunato(2011) Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84:066122. Cited in page(s) [43](#)

Lancichinetti *et al.*(2008) Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110(1–5). Cited in page(s) [29](#), [85](#), [117](#), [182](#), [184](#), [186](#)

Lancichinetti *et al.*(2009) Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015. Cited in page(s) [10](#), [28](#), [84](#)

Latapy *et al.*(2008) Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31–48. Cited in page(s) [30](#)

LeCun *et al.*(1998) Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Cited in page(s) [121](#), [123](#), [138](#), [234](#)

Lee and Verleysen(2007) John A. Lee and Michel Verleysen. *Nonlinear Dimensionality Reduction*. Springer. Cited in page(s) [201](#)

Liang *et al.*(2009) Jinling Liang, Zidong Wang, and Xiaohui Liu. State estimation for coupled uncertain stochastic networks with missing measurements and time-varying delays: The discrete-time case. *IEEE Trans. Neural Networks*, 20(5):781–793. Cited in page(s) [2](#)

- Liben-Nowell and Kleinberg(2007)** David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031. Cited in page(s) [47](#)
- Lim and Park(2009)** Gaksoo Lim and Cheong Hee Park. Semi-supervised dimension reduction using graph-based discriminant analysis. In *CIT 1*, pages 9–13. IEEE Computer Society. Cited in page(s) [35](#)
- Lima et al.(2001)** Gilson F. Lima, Alexandre S. Martinez, and Osame Kinouchi. Deterministic walks in random media. *Physical Review Letters*, 87:010603. Cited in page(s) [33](#), [34](#), [213](#)
- Lin and Wang(2002)** Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13:464–471. Cited in page(s) [218](#), [231](#)
- Liu et al.(2002)** Cheng-Lin Liu, Hiroshi Sako, and Hiromichi Fujisawa. Performance evaluation of pattern classifiers for handwritten character recognition. *IJDAR*, 4: 191–204. Cited in page(s) [133](#)
- Liu et al.(2008)** Derong Liu, Zhongyu Pang, and Stephen R. Lloyd. A neural network method for detection of obstructive sleep apnea and narcolepsy based on pupil size and EEG. *IEEE Transactions on Neural Networks*, 19(2):308–318. Cited in page(s) [2](#)
- Liu et al.(2012)** Derong Liu, Chin-Teng Lin, Garry Greenwood, Simon Lucas, and Zhengyou Zhang. CIS Publication Spotlight - Publication Spotlight. *IEEE Computational Intelligence Magazine*, 7(3):11–12. doi: <http://dx.doi.org/10.1109/MCI.2012.2200619>. URL <http://dblp.uni-trier.de/db/journals/cim/cim7.html#LiuLGLZ12b>. Cited in page(s) [140](#), [257](#)
- Liu et al.(2004)** Hancong Liu, Sirish Shah, and Wei Jiang. On-line outlier detection and data cleaning. *28th Computers and Chemical Engineering*, pages 1635–1647. Cited in page(s) [35](#)
- Liu et al.(2010)** Jialu Liu, Deng Cai, and Xiaofei He. Gaussian mixture model with local consistency. In *AAAI'10*, volume 1, pages 512–517. Cited in page(s) [136](#), [137](#)
- López-Rubio et al.(2009)** Ezequiel López-Rubio, Juan M. Ortiz de Lazcano-Lobato, and Domingo López-Rodríguez. Probabilistic PCA self-organizing maps. *IEEE Transactions on Neural Networks*, 20(9):1474–1489. Cited in page(s) [2](#)
- Lu et al.(2003)** Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Algorithms for spatial outlier detection. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*. IEEE Computer Society. Cited in page(s) [35](#)
- Lu and Weng(2007)** Dengsheng Lu and Qihao Weng. Survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870. Cited in page(s) [202](#)
- Lu and Getoor(2003)** Qing Lu and Lise Getoor. Link-based classification using labeled and unlabeled data. In *International Conference on Machine Learning Workshop on "The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining"*, page 496–503. Cited in page(s) [47](#), [48](#)

Lu and Ip(2009) Zhiwu Lu and Horace H. S. Ip. Generalized competitive learning of gaussian mixture models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(4):901–909. Cited in page(s) [2](#)

Lusseau(2003) David Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society B: Biological Sciences*, 270 Suppl 2:S186–S188. Cited in page(s) [128](#)

MacQueen(1967) James B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press. Cited in page(s) [38, 39, 136](#)

Macskassy and Provost(2003) Sofus A. Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at the Knowledge Discovery and Data Mining Conference (KDD)*, pages 64–76. Cited in page(s) [47](#)

Macskassy and Provost(2007) Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983. Cited in page(s) [48, 51, 202](#)

Maier and von Luxburg(2009) Markus Maier and Ulrike von Luxburg. Influence of graph construction on graph-based clustering measures. *The Neural Information Processing Systems*, 22:1025–1032. Cited in page(s) [204](#)

McDowell et al.(2009) Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious collective classification. *Journal of Machine Learning Research*, 10:2777–2836. Cited in page(s) [48, 49](#)

Meyer-Bäse and Thümmler(2008) Anke Meyer-Bäse and Vera Thümmler. Local and global stability analysis of an unsupervised competitive neural network. *IEEE Transactions on Neural Networks*, 19(2):346–351. Cited in page(s) [2](#)

Meyn and Tweedie(2009) Sean P. Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, second edition. Cited in page(s) [22](#)

Micheli(2009) Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511. Cited in page(s) [202](#)

Milgram(1967) Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67. Cited in page(s) [19](#)

Mitchell(1997) Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, first edition. Cited in page(s) [1, 35, 36, 38, 39, 44](#)

Mizruchi(1982) Mark S. Mizruchi. The american corporate network. *Sage*, 2:1904–1974. Cited in page(s) [20](#)

Montoya and Solée(2002) Jose M. Montoya and Ricard V. Solée. Small world patterns in food webs. *Journal of Theoretical Biology*, 214:405–412. Cited in page(s) [20](#)

- Mori *et al.*(1992)** Shunji Mori, Ching Y. Suen, and Kazuhiko Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE*, 80:1029–1058. Cited in page(s) [133](#)
- Neapolitan(2003)** Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, first edition. Cited in page(s) [46](#), [231](#)
- Newman(2010)** M. E. J. Newman. *Networks: An Introduction*. Oxford University Press. Cited in page(s) [3](#), [31](#), [32](#)
- Newman(2002)** Mark. E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701. Cited in page(s) [30](#)
- Newman(2003a)** Mark E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126. Cited in page(s) [30](#)
- Newman(2004)** Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133. Cited in page(s) [4](#), [28](#), [40](#), [120](#), [127](#)
- Newman(2006a)** Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582. Cited in page(s) [4](#), [26](#), [32](#), [35](#), [59](#), [129](#), [131](#)
- Newman(2006b)** Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104. Cited in page(s) [27](#), [35](#), [40](#)
- Newman(2003b)** Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256. Cited in page(s) [3](#), [18](#), [20](#), [22](#), [131](#)
- Newman and Leicht(2007)** Mark E. J. Newman and Elizabeth A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104(23):9564–9569. Cited in page(s) [27](#)
- Newman and Girvan(2004)** Michele E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review Letters*, (69):026113. Cited in page(s) [4](#), [9](#), [26](#), [27](#), [28](#), [38](#), [40](#), [80](#)
- Nicosia *et al.*(2009)** Vincenzo Nicosia, Giuseppe Mangioni, Vicenza Carchiolo, and Michele Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, 2009 (03):03024. Cited in page(s) [10](#), [28](#), [84](#)
- Nigam *et al.*(2000)** Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134. Cited in page(s) [55](#)
- Noh and Rieger(2004)** Jae Dong Noh and Heiko Rieger. Random walks on complex networks. *Physical Review Letters*, 92:118701. Cited in page(s) [2](#)
- Olaru(2003)** Cristina Olaru. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221–254. Cited in page(s) [231](#)
- Opsahl and Panzarasa(2009)** Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163. Cited in page(s) [30](#)

Palla *et al.*(2005) Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818. Cited in page(s) [3](#), [10](#), [11](#), [27](#), [28](#), [80](#), [84](#)

Pearson(1905) Karl Pearson. The problem of the random walk. *Nature*, 72(1867):294. Cited in page(s) [2](#)

Piatetsky-Shapiro(1991) Gregory Piatetsky-Shapiro. *Discovery, Analysis, and Presentation of Strong Rules*. AAAI/MIT Press, Cambridge, MA. Cited in page(s) [35](#)

Pradeep *et al.*(2011) Jayabala Pradeep, E. Srinivasan, and S. Himavathi. Diagonal based feature extraction for handwritten alphabets recognition system using neural network. *International Journal of Computer Science and Information Technology*, 3:27–38. Cited in page(s) [133](#)

Príncipe and Miikkulainen(2009) José C. Príncipe and Risto Miikkulainen. *Advances in Self-Organizing Maps - 7th International Workshop, WSOM 2009, Lecture Notes in Computer Science*, Vol. 5629. Springer. Cited in page(s) [2](#)

Quiles *et al.*(2008) Marcos G. Quiles, Liang Zhao, Ronaldo L. Alonso, and Roseli A. F. Romero. Particle competition for complex network community detection. *Chaos*, 18(3):033107. Cited in page(s) [9](#), [40](#), [42](#), [109](#), [118](#), [119](#), [120](#), [253](#), [254](#)

Quinlan(1987) J. Ross Quinlan. Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, page 304–307. Cited in page(s) [45](#)

Quinlan(1992) J. Ross Quinlan. *C4.5: Programs for Machine Learning* (Morgan Kaufmann Series in Machine Learning). First edition. Cited in page(s) [45](#)

Ratle *et al.*(2008) Frédéric Ratle, Jason Weston, and Matthew L. Miller. Large-scale clustering through functional embedding. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), pages 266–281. Springer-Verlag. Cited in page(s) [136](#), [137](#)

Ravasz *et al.*(2002) Erzsébet Ravasz, Anna L. Somera, Dale A. Mongru, Zoltan N. Oltvai, and Albert-László Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555. Cited in page(s) [26](#)

Reichardt and Bornholdt(2004) Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21):218701(1–4). Cited in page(s) [4](#), [9](#), [28](#), [40](#)

Rosvall and Bergstrom(2007) Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331. Cited in page(s) [27](#)

Rumelhart *et al.*(1988) David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*, pages 673–695. MIT Press, Cambridge, MA, USA. Cited in page(s) [231](#)

- Scott(2000)** John P. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, first edition. Cited in page(s) [20](#)
- Segal et al.(2003)** Eran Segal, Haidong Wang, and Daphne Koller. Discovering molecular pathways from protein interaction and gene expression data. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 264–272. Cited in page(s) [47](#)
- Sen et al.(2008)** Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *Artificial Intelligence Magazine*, 29(3):93–106. Cited in page(s) [48](#)
- Seung and Lee(2000)** H. Sebastian Seung and Daniel D. Lee. The manifold ways of perception. *Science*, 290(5500):2268–2269. Cited in page(s) [201](#)
- Shadbolt et al.(2006)** Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 6:96–101. Cited in page(s) [201](#)
- Shanmugam et al.(2010)** Geetha Shanmugam, Poonthalir Ganesan, and Vanathi P. T. A hybrid particle swarm optimization with genetic operators for vehicle routing problem. *Journal of Advances in Information Technology*, 1(4):1693–1702. Cited in page(s) [81](#)
- Shen et al.(2009)** Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8):1706 – 1712. Cited in page(s) [10](#), [28](#), [84](#)
- Shi and Malik(1997)** Jianbo Shi and Jitendra Malik. Normalized cut and image segmentation. Technical report, Berkeley, CA, USA. Cited in page(s) [136](#)
- Shi and Eberhart(1998)** Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation*, pages 69–73. Cited in page(s) [231](#)
- SIBGRAPI(2011)** SIBGRAPI. Best paper awards. *Graphics, Patterns and Images, SIBGRAPI Conference on*, 0:xxii–xxiii. ISSN 1530-1834. doi: <http://doi.ieeecomputersociety.org/10.1109/SIBGRAPI.2011.59>. Cited in page(s) [140](#)
- Silva and Amancio(2012)** Thiago C. Silva and Diego R. Amancio. Word sense disambiguation via high order of learning in complex networks. *Europhysics Letters*, 98: 58001. Cited in page(s) [257](#)
- Silva and Zhao(2012a)** Thiago C. Silva and Liang Zhao. Stochastic competitive learning in complex networks. *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):385–398. doi: [10.1109/TNNLS.2011.2181866](https://doi.org/10.1109/TNNLS.2011.2181866). Cited in page(s) [5](#), [140](#), [202](#), [257](#)
- Silva and Zhao(2012b)** Thiago C. Silva and Liang Zhao. Network-based stochastic semisupervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):451–466. doi: [10.1109/TNNLS.2011.2181413](https://doi.org/10.1109/TNNLS.2011.2181413). Cited in page(s) [7](#), [202](#), [257](#)
- Silva and Zhao(2012c)** Thiago C. Silva and Liang Zhao. Semi-supervised learning guided by the modularity measure in complex networks. *Neurocomputing*, 78(1):30 – 37. Cited in page(s) [57](#), [59](#), [60](#), [193](#), [202](#), [257](#)

- Silva and Zhao(2012d)** Thiago C. Silva and Liang Zhao. Network-based high level data classification. *IEEE Transactions on Neural Networks and Learning Systems*, 23(6): 954–970. doi: 10.1109/TNNLS.2012.2195027. Cited in page(s) [7](#), [257](#)
- Silva and Zhao(2012e)** Thiago C. Silva and Liang Zhao. Detecting error propagation via competitive learning. *Procedia Computer Science*, 13:37–42. Cited in page(s) [258](#)
- Silva and Zhao(2012f)** Thiago C. Silva and Liang Zhao. Detecting and preventing error propagation via competitive learning. *Neural Networks*. doi: 10.1016/j.neunet.2012.11.001. Cited in page(s) [7](#), [257](#)
- Silva and Zhao(2012g)** Thiago C. Silva and Liang Zhao. Pattern recognition using complex networks. In *Workshop of Theses and Dissertations (WTD) in SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images)*, pages 130–135. Cited in page(s) [258](#)
- Silva and Zhao(2011a)** Thiago C. Silva and Liang Zhao. Network-based learning through particle competition for data clustering. In *2011 International Joint Conference on Neural Networks*, pages 45–52. Cited in page(s) [258](#)
- Silva and Zhao(2011b)** Thiago C. Silva and Liang Zhao. Semisupervised learning in complex networks. In *XXXI Congresso da Sociedade Brasileira de Computação, ENIA*, pages 406–417. Cited in page(s) [258](#)
- Silva and Zhao(2011c)** Thiago C. Silva and Liang Zhao. Uncovering overlapping structures via stochastic competitive learning. In *DINCON - 10a Conferência Brasileira de Dinâmica, Controle e Aplicações*, volume 1, pages 47–50. Cited in page(s) [258](#)
- Silva et al.(2011a)** Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao. Stochastic competitive learning applied to handwritten digit and letter clustering. In *Proceedings of the XXIV Sibgrapi Conference on Graphics, Patterns and Images*, pages 313–320. Cited in page(s) [140](#), [258](#)
- Silva et al.(2011b)** Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao. High level classification for pattern recognition. In *Proceedings of the XXIV Sibgrapi Conference on Graphics, Patterns and Images*, pages 344–351. Cited in page(s) [258](#)
- Silva et al.(2012a)** Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao. Detecting overlapping structures via network-based competitive learning. In *WCCI 2012 IEEE World Congress on Computational Intelligence, Proceedings of the International Joint Conference on Neural Networks (IJCNN 2012)*, pages 3036–3043. Cited in page(s) [258](#)
- Silva et al.(2012b)** Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao. Preventing error propagation in semisupervised learning. In *Proceedings of The Ninth International Symposium on Neural Networks (ISNN 2012), Lecture Notes in Computer Science. Lecture Notes in Computer Science*, volume 7367, pages 565–572. Cited in page(s) [258](#)
- Silva et al.(2012c)** Thiago C. Silva, Liang Zhao, and Thiago H. Cupertino. Handwritten data clustering using agents competition in networks. *Journal of Mathematical Imaging and Vision*. doi: 10.1007/s10851-012-0353-z. Cited in page(s) [5](#), [257](#)
- Sindhwani et al.(2005)** Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the*

- 22nd international conference on Machine learning (ICML)**, pages 824–831, New York, NY, USA. ACM Press. Cited in page(s) [47](#), [171](#)
- Singh et al.(2008)** Aarti Singh, Robert D. Nowak, and Xiaojin Zhu. Unlabeled data: Now it helps, now it doesn't. In *The Conference on Neural Information Processing Systems NIPS*, pages 1513–1520. Cited in page(s) [2](#), [53](#)
- Skolidis and Sanguinetti(2011)** Grigorios Skolidis and Guido Sanguinetti. Bayesian multitask classification with gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12):2011–2021. Cited in page(s) [202](#)
- Sporns(2002)** Olaf Sporns. Networks analysis, complexity, and brain function. *Complexity*, 8(1):56–60. Cited in page(s) [20](#)
- Stanley and Buldyrev(2001)** Harry Eugene Stanley and Sergey V. Buldyrev. Statistical physics - the salesman and the tourist. *Nature*, 413:373–374. Cited in page(s) [34](#), [213](#)
- Strogatz(2001)** Steven H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276. Cited in page(s) [20](#)
- Sugar and James(2003)** Catherine A. Sugar and Gareth M. James. Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association*, 98:750–763. Cited in page(s) [64](#)
- Sun et al.(2006)** Jun Sun, Stephen Boyd, Lin Xiao, and Persi Diaconis. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 48:681–699. ISSN 0036-1445. Cited in page(s) [171](#)
- Sun et al.(2011)** Peng Gang Sun, Lin Gao, and Shan Shan Han. Identification of overlapping and non-overlapping community structure by fuzzy clustering in complex networks. *Information Sciences*, 181:1060–1071. Cited in page(s) [10](#), [28](#)
- Szummer and Jaakkola(2001)** Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, pages 945–952. Cited in page(s) [57](#)
- Tan et al.(2008)** Ah-Hwee Tan, Ning Lu, and Dan Xiao. Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *IEEE Transactions on Neural Networks*, 19(2):230–244. Cited in page(s) [2](#)
- Tenenbaum et al.(2000)** Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. Cited in page(s) [201](#)
- Theodoridis and Koutroumbas(2008)** Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, fourth edition. Cited in page(s) [81](#), [133](#)
- Tian et al.(2000)** Bin Tian, Mahmood R. Azimi-Sadjadi, Thomas H. V. Haar, and Donald Reinke. Temporal updating scheme for probabilistic neural network with application to satellite cloud classification. *IEEE Transactions on Neural Networks*, 11(4):903–920. Cited in page(s) [202](#)

Tian et al.(2006) Yonghong Tian, Qiang Yang, Tiejun Huang, Charles X. Ling, and Wen Gao. Learning contextual dependency network models for link-based classification. *IEEE Transactions on Data and Knowledge Engineering*, 18(11):1482–1496. Cited in page(s) 202

Tim Berners-Lee and and Lassila(2001) James Hendler Tim Berners-Lee and and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43. Cited in page(s) 201

Tsai et al.(2010) S.-H. Tsai, C.-Y. Lee, and Y.-K. Wu. Efficient calculation of critical eigenvalues in large power systems using the real variant of the Jacobi-Davidson QR method. *Generation, Transmission and Distribution, IET*, 4:467–478. Cited in page(s) 134

Tuia et al.(2010) Devis Tuia, Gustavo Camps-Valls, Giona Matasci, and Mikhail Kanevski. Learning relevant image features with multiple-kernel classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10):3780–3791. Cited in page(s) 202

van der Merwe and Engelbrecht(2003) Dean W. van der Merwe and Andries P. Engelbrecht. Data clustering using particle swarm optimization. *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, 1:215–220. Cited in page(s) 123

Vapnik(1995) Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. Cited in page(s) 46

Vapnik(1998) Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, first edition. Cited in page(s) 55, 171, 218

Wang et al.(2009) Chao-Huang Wang, Chung-Nan Lee, and Chaur-Heh Hsieh. *Variants of Self-Organizing Maps: Applications in Image Quantization and Compression*. Lambert Academic Publishing. Cited in page(s) 2, 64

Wang and Zhang(2008) Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67. Cited in page(s) 57, 171, 193

Wang et al.(2008) Fei Wang, Tao Li, Gang Wang, and Changshui Zhang. Semi-supervised classification using local and global regularization. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 726–731. AAAI Press. Cited in page(s) 54, 57

Wang(1997) Lipo Wang. On competitive learning. *IEEE Transactions on Neural Networks*, 8(5):1214–1217. Cited in page(s) 127

Watts(2003) Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness (Princeton Studies in Complexity)*. Princeton University Press, first edition. Cited in page(s) 22, 23

Watts and Strogatz(1998) Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442. Cited in page(s) 3, 19, 22, 23, 30

Weinberger and Saul(2006) Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90. ISSN 0920-5691. Cited in page(s) 171

- West et al.(1999)** Geoffrey B. West, James H. Brown, and Brian J. Enquist. A general model for the structure, and algometry of plant vascular systems. *Nature*, 400:122–126. Cited in page(s) [20](#)
- Williams et al.(2007)** David Williams, Xuejun Liao, Ya Xue, and Lawrence Carin. On classification with incomplete data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):427–436. Cited in page(s) [202](#)
- Wu and Huberman(2004)** Fang Wu and Bernardo A. Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):331–338. Cited in page(s) [79](#)
- Wu and Schölkopf(2007)** Mingrui Wu and Bernhard Schölkopf. Transductive classification via local learning regularization. In *11th International Conference on Artificial Intelligence and Statistics*, pages 628–635. Microtome. Cited in page(s) [56](#)
- Xu and II(2005)** Rui Xu and Donald Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678. Cited in page(s) [2](#), [38](#), [39](#)
- Zachary(1977)** Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473. Cited in page(s) [120](#), [127](#)
- Zeng et al.(2010)** Yingpei Zeng, Jiannong Cao, Shigeng Zhang, Shanqing Guo, and Li Xie. Random-walk based approach to detect clone attacks in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(5):677–691. Cited in page(s) [2](#), [171](#)
- Zhang and Mao(2008)** Dell Zhang and Robert Mao. Classifying networked entities with modularity kernels. In *International Conference on Information and Knowledge Management*, pages 113–122. Cited in page(s) [202](#)
- Zhang et al.(2011)** Huaguang Zhang, Jinhai Liu, Dazhong Ma, and Zhanshan Wang. Data-core-based fuzzy min-max neural network for pattern classification. *IEEE Transactions on Neural Networks*, 22(12):2339–2352. Cited in page(s) [202](#)
- Zhang et al.(2007)** Shihua Zhang, Rui-Sheng Wang, and Xiang-Sun Zhang. Identification of overlapping community structure in complex networks using fuzzy C-Means clustering. *Physica A: Statistical Mechanics and its Applications*, 374(1):483–490. Cited in page(s) [10](#), [28](#), [84](#)
- Zhang et al.(2006)** Tong Zhang, Alexandrin Popescul, and Byron Dom. Linear prediction models with graph regularization for web-page categorization. In *Conference on Knowledge Discovery and Data Mining*, pages 821–826. ACM. Cited in page(s) [202](#)
- Zhao et al.(2004)** Liang Zhao, Kwangho Park, and Ying-Cheng Lai. Attack vulnerability of scale-free networks due to cascading breakdown. *Physical Review E*, 70:035101(1–4). Cited in page(s) [20](#)
- Zhao et al.(2005)** Liang Zhao, Kwangho Park, and Ying-Cheng Lai. Tolerance of scale-free networks against attack-induced cascades. *Physical Review E (Rapid Communication)*, 72(2):025104(R)1–4. Cited in page(s) [20](#)

- Zhao et al.(2007)** Liang Zhao, Thiago H. Cupertino, Kwangho Park, Ying-Cheng Lai, and Xiaogang Jin. Optimal structure of complex networks for minimizing traffic congestion. *Chaos (Woodbury)*, 17(4):043103(1–5). Cited in page(s) [20](#)
- Zhong et al.(2008)** Ming Zhong, Kai Shen, and Joel Seiferas. The convergence-guaranteed random walk and its applications in peer-to-peer networks. *IEEE Transactions on Computers*, 57(5):619–633. Cited in page(s) [2](#)
- Zhou and Schölkopf(2006)** Dengyong Zhou and Bernhard Schölkopf. *Semi-supervised Learning. ch: Discrete Regularization*, pages 237–250. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA. Cited in page(s) [171](#)
- Zhou et al.(2004)** Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, volume 16, pages 321–328. MIT Press. Cited in page(s) [10](#), [55](#), [56](#), [57](#), [58](#), [171](#), [192](#), [193](#)
- Zhou(2003a)** Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(6):061901. Cited in page(s) [2](#), [4](#), [28](#), [40](#)
- Zhou(2003b)** Haijun Zhou. Network landscape from a brownian particle's perspective. *Physical Review E*, 67(4):041908. Cited in page(s) [9](#)
- Zhu et al.(2007)** Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *Special Interest Group on Information Retrieval*, pages 487–494. ACM. Cited in page(s) [202](#)
- Zhu(2005a)** Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. Cited in page(s) [56](#), [57](#), [202](#)
- Zhu(2005b)** Xiaojin Zhu. Semi-supervised learning with graphs. *Doctoral Thesis - Carnegie Mellon University. CMU-LTI-05-192*. Cited in page(s) [56](#)
- Zhu and Ghahramani(2002)** Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, Pittsburgh. Cited in page(s) [57](#), [171](#)
- Zhu and Goldberg(2009)** Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, Synthesis Lectures on Artificial Intelligence and Machine Learning. Cited in page(s) [55](#)
- Zhu et al.(2003)** Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, pages 912–919. Cited in page(s) [56](#), [57](#)