
Machine learning via dynamical processes in complex
networks

Thiago Henrique Cupertino

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Machine learning via dynamical processes in complex networks

Thiago Henrique Cupertino

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. *FINAL VERSION*.

USP – São Carlos
February 2014

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

C974m Cupertino, Thiago Henrique
Machine learning via dynamical processes on
complex networks / Thiago Henrique Cupertino;
orientador Zhao Liang. -- São Carlos, 2013.
175 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, 2013.

1. Machine learning. 2. Complex networks. 3.
Dynamical processes. 4. Classification. 5.
Clustering. I. Liang, Zhao, orient. II. Título.

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Aprendizado de máquina via processos dinâmicos em redes complexas

Thiago Henrique Cupertino

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. *VERSÃO REVISADA.*

USP – São Carlos
Fevereiro de 2014

Acknowledgements

My decisive contact with research started when I first came to the University of São Paulo in São Carlos in 2004 to begin my undergraduate studies at the Institute of Mathematical and Computer Sciences and at the School of Engineering. There, I found a land of opportunities to carry out cutting edge researches.

I would like to thank all the mates I had the pleasure to live with in our communal houses during so many years in São Carlos. Our fellowship was significant for all of us to carry on our studies and to overcome the challenges each of us would inevitably face one day or another. I made so many real friends that I never had the chance to make before. Those days are unforgettable.

I would also like to thank all the friends I made during the postgraduate period. We were able to share issues from our research to help each other; we spent really good moments together through barbecues, lunch at the university restaurant, coffee breaks, conferences and writing papers; the discussions at the lab were really fruitful.

I thank the university staff for being so helpful and solicitous to all the students, who, undoubtedly, always needed their help.

I also thank Fapesp for the financial support, which was fundamental to carry out all the research activities.

I would like to thank the professors for instructing me with excellence techniques, extensive knowledge and education on uncountable lectures. This support has made me the professional I am today and helped me to accomplish many objectives.

I would also like to thank professor Zhao Liang for being my tutor from 2005, when we started undergraduate research, until today, when I am writing this Ph.D. thesis. He has shown to me the real value of working to chase our objectives in research. I appreciate his patience and perseverance. He supported me with words of wisdom when things seemed out of reach and when I needed his invaluable advices.

I would like to thank my sweetie, Liana, who has been with me during the last years. She has been patient, comprehensive and kind even in distressing moments. The great times we spend together soothe me and I can carry on with the challenges in life.

I would like to thank my siblings, Antonio and Hellen. We know that we have always been there to help each other even though living far apart to pursue our dreams.

I would also like to thank my parents to whom I will be eternally grateful for raising me with the best they could. Mom and Dad, thanks for always believing in me and supporting my endeavors.

Abstract

Extracting useful knowledge from data sets is a key concept in modern information systems. Consequently, the need of efficient techniques to extract the desired knowledge has been growing over time. Machine learning is a research field dedicated to the development of techniques capable of enabling a machine to "learn" from data. Many techniques have been proposed so far, but there are still issues to be unveiled specially in interdisciplinary research. In this thesis, we explore the advantages of network data representation to develop machine learning techniques based on dynamical processes on networks. The network representation unifies the structure, dynamics and functions of the system it represents, and thus is capable of capturing the spatial, topological and functional relations of the data sets under analysis. We develop network-based techniques for the three machine learning paradigms: supervised, semi-supervised and unsupervised. The random walk dynamical process is used to characterize the access of unlabeled data to data classes, configuring a new heuristic we call ease of access in the supervised paradigm. We also propose a classification technique which combines the high-level view of the data, via network topological characterization, and the low-level relations, via similarity measures, in a general framework. Still in the supervised setting, the modularity and Katz centrality network measures are applied to classify multiple observation sets, and an evolving network construction method is applied to the dimensionality reduction problem. The semi-supervised paradigm is covered by extending the ease of access heuristic to the cases in which just a few labeled data samples and many unlabeled samples are available. A semi-supervised technique based on interacting forces is also proposed, for which we provide parameter heuristics and stability analysis via a Lyapunov function. Finally, an unsupervised network-based technique uses the concepts of pinning control and consensus time from dynamical processes to derive a similarity measure used to cluster data. The data is represented by a connected and sparse network in which nodes are dynamical elements. Simulations on benchmark data sets and comparisons to well-known machine learning techniques are provided for all proposed techniques. Advantages of network data representation and dynamical processes for machine learning are highlighted in all cases.

Keywords: machine learning, supervised learning, semi-supervised learning, unsupervised learning, dimensionality reduction, network-based learning, complex networks, dynamical processes, random walk, limiting probabilities, stationary states, consensus time, pinning control, interacting forces.

Resumo

A extração de conhecimento útil a partir de conjuntos de dados é um conceito chave em sistemas de informação modernos. Por conseguinte, a necessidade de técnicas eficientes para extrair o conhecimento desejado vem crescendo ao longo do tempo. Aprendizado de máquina é uma área de pesquisa dedicada ao desenvolvimento de técnicas capazes de permitir que uma máquina "aprenda" a partir de conjuntos de dados. Muitas técnicas já foram propostas, mas ainda há questões a serem reveladas especialmente em pesquisas interdisciplinares. Nesta tese, exploramos as vantagens da representação de dados em rede para desenvolver técnicas de aprendizado de máquina baseadas em processos dinâmicos em redes. A representação em rede unifica a estrutura, a dinâmica e as funções do sistema representado e, portanto, é capaz de capturar as relações espaciais, topológicas e funcionais dos conjuntos de dados sob análise. Desenvolvemos técnicas baseadas em rede para os três paradigmas de aprendizado de máquina: supervisionado, semissupervisionado e não supervisionado. O processo dinâmico de passeio aleatório é utilizado para caracterizar o acesso de dados não rotulados às classes de dados configurando uma nova heurística no paradigma supervisionado, a qual chamamos de facilidade de acesso. Também propomos uma técnica de classificação de dados que combina a visão de alto nível dos dados, por meio da caracterização topológica de rede, com relações de baixo nível, por meio de medidas de similaridade, em uma estrutura geral. Ainda no aprendizado supervisionado, as medidas de rede modularidade e centralidade Katz são aplicadas para classificar conjuntos de múltiplas observações, e um método de construção evolutiva de rede é aplicado ao problema de redução de dimensionalidade. O paradigma semissupervisionado é abordado por meio da extensão da heurística de facilidade de acesso para os casos em que apenas algumas amostras de dados rotuladas e muitas amostras não rotuladas estão disponíveis. É também proposta uma técnica semissupervisionada baseada em forças de interação, para a qual fornecemos heurísticas para selecionar parâmetros e uma análise de estabilidade mediante uma função de Lyapunov. Finalmente, uma técnica não supervisionada baseada em rede utiliza os conceitos de controle pontual e tempo de consenso de processos dinâmicos para derivar uma medida de similaridade usada para agrupar dados. Os dados são representados por uma rede conectada e esparsa na qual os vértices são elementos dinâmicos. Simulações com dados de referência e comparações com técnicas de aprendizado de máquina conhecidas são fornecidas para todas as técnicas propostas. As vantagens da representação de dados em rede e de processos dinâmicos para o aprendizado de máquina são evidenciadas em todos os casos.

Palavras-chave: aprendizado de máquina, aprendizado supervisionado, aprendizado semissupervisionado, aprendizado não supervisionado, redução de dimensionalidade, aprendizado baseado em redes, redes complexas, processos dinâmicos, caminhada aleatória, probabilidades limite, estado estacionário, tempo de consenso, controle pontual, forças de interação.

Contents

Acknowledgements	vii
Abstract	ix
Resumo	xi
Contents	xiii
List of figures	xvii
List of tables	xxi
List of algorithms	xxiii
List of symbols	xxv
1 Introduction	1
1.1 Objectives	5
1.2 Motivations	7
1.3 Organization of this thesis	11
2 Review of essential concepts	13
2.1 Machine learning	14
2.1.1 Supervised learning	15
2.1.2 Semi-supervised learning	18
2.1.3 Unsupervised learning	21
2.2 Complex networks	23
2.2.1 Basics from graph theory	25
2.2.2 Network measures	27
2.2.3 Constructing networks from data sets	31
2.3 Dynamical processes on networks	32
2.3.1 Random walk	32
2.3.2 Tourist walk	36
2.3.3 Chaotic synchronization	38
2.3.4 Consensus and pinning control	39
2.4 Relevant network-based techniques in machine learning	40

2.4.1	Particle competition for unsupervised and semi-supervised learning	41
2.4.2	Unsupervised scene segmentation via synchronization of locally coupled chaotic oscillators	45
2.4.3	Semi-supervised label propagation: local and global consistency	47
2.4.4	Supervised k-associated graphs	51
3	Development of network-based techniques for supervised learning	53
3.1	Measuring ease of access to data classes by using random walk	55
3.1.1	Technique description	55
3.1.2	Algorithm and complexity analysis	58
3.1.3	Experimental results	59
3.2	High-level inductive classification	65
3.2.1	The general classification framework	67
3.2.1.1	Similarity weight matrix	69
3.2.1.2	Structural weight matrix	69
3.2.2	Algorithm and complexity analysis	71
3.2.3	Experimental results	72
3.2.3.1	Toy problems	72
3.2.3.2	Effectiveness of the structural term	76
3.2.3.3	Influence of parameter α	77
3.2.3.4	Influence of parameter τ	79
3.2.3.5	Comparative study	81
3.2.3.6	Application to image classification	86
3.3	Classification of multiple observation sets	90
3.3.1	The problem of multiple observation sets	90
3.3.2	Classification via modularity measure	90
3.3.3	Classification via Katz index	92
3.3.4	Algorithm and complexity analysis	92
3.3.5	Experimental results	93
3.4	kAOG embedding for dimensionality reduction	97
3.4.1	The Yan's graph-preserving criterion	98
3.4.2	kAOG into the graph-embedding framework	99
3.4.3	Experimental results	101
4	Development of network-based techniques for semi-supervised learning	107
4.1	Semi-supervised transductive classification via random walk limiting probabilities	108
4.1.1	Technique description	108
4.1.2	Algorithm and complexity analysis	111
4.1.3	Experimental results	112
4.1.3.1	Toy example	112
4.1.3.2	Benchmark data sets	113
4.2	Semi-supervised classification based on interacting forces	115
4.2.1	Technique description	116
4.2.2	Algorithm	119
4.2.3	Stability analysis	120
4.2.4	Parameter analysis	121

4.2.4.1	Parameters β and δ	121
4.2.4.2	Parameter ϕ	122
4.2.5	Experimental results	125
5	Development of network-based techniques for unsupervised learning	135
5.1	Data clustering via consensus dissimilarity	136
5.1.1	Dissimilarity measure based on consensus time	136
5.1.2	Convergence analysis	137
5.1.3	Technique description	141
5.1.3.1	Network construction	141
5.1.3.2	Cluster detection	143
5.1.4	Experimental results	144
5.1.4.1	Influence of parameter k	146
5.1.4.2	Comparisons	146
6	Conclusions	151
6.1	Conclusions and future studies	152
6.2	List of publications	156
	Bibliography	161

List of figures

2.1	Illustration of training and test phases in supervised classification	16
2.2	Illustration of semi-supervised classification	19
2.3	Illustration of data clustering process	22
2.4	A small network to illustrate the clustering coefficient measure.	28
2.5	Examples of networks constructed by using different methods	33
2.6	Transition graph of the Markov matrix in Eq. 2.11	35
2.7	Illustration of a tourist walk route	37
2.8	Illustration of community detection via particle competition in an artificial network	44
2.9	Sample image for segmentation	46
2.10	Segmentation result when Fig. 2.9 is an input	46
2.11	Temporal activities of oscillator blocks when Fig. 2.9 is an input.	46
2.12	The two-moons problem to illustrate semi-supervised classification . . .	48
2.13	Calculation of the purity measure in 3-nearest neighbors networks . . .	52
3.1	Illustration to calculate the modified connection matrix $\hat{\mathcal{W}}$ when classifying the unlabeled instance \mathbf{x}^u	57
3.2	Classification accuracy in function of parameter ϵ for the simulated data sets	61
3.3	Similarity values among network nodes used in the matrix composition \mathcal{W}_{sim} of the toy example	73
3.4	Artificial data set composed of two classes forming specific patterns . . .	74
3.5	Correct classification for the data set in Fig. 3.4	75
3.6	Structural changes caused by the insertion of test instances on each class in Fig. 3.4	76
3.7	Classification accuracy in function of parameter α for Iris, Segment and Vehicle data sets	78
3.8	Class distribution of the Iris data set using the first three most representatives PCA components	79
3.9	Class distribution of the Segment data set using the first three most representatives PCA components	80
3.10	Class distribution of the Segment data set using the first three most representatives PCA components	80
3.11	Data set class dispersion and the optimal α values for all data sets in Table 3.1	81
3.12	Classification accuracy in function of parameter τ	82

3.13	ETH-80 images collection	87
3.14	The 41 instances of the yellow race car class in the ETH-80 images collection	87
3.15	Classification accuracy of the categories Cow, Cup and Pear, in function of parameter α	88
3.16	An example of a k-NN patterns network formation	91
3.17	Examples from the handwritten digit images data set	95
3.18	Classification error rates (%) using different values for parameter k of network construction on the handwritten digits data set	95
3.19	Classification error rates (%) for the handwritten digit images data set	96
3.20	Results of the proposed dimensionality reduction technique for the 5 data sets and comparison to the original attribute space size.	103
3.21	Classification accuracy by using different numbers of projected dimensions	104
3.22	Image examples from the <i>binary alphasdigits</i> data set	105
3.23	Classification accuracy on images of numbers from <i>binary alphasdigits</i> data set in function of the number of transformed attributes used in classification	105
3.24	Classification accuracy on all images available into <i>binary alphasdigits</i> data set in function of the number of transformed attributes used in classification	106
4.1	Composition process of the modified connection matrix $\hat{\mathcal{W}}_j^{(l)}$	110
4.2	Mix of different cluster shapes from artificial data for semi-supervised classification	113
4.3	Classification achieved by the proposed technique for the data set depicted in Fig. 4.2.	114
4.4	2-dimensional visualization of the force function shape showing its maximum amplitudes at points y^* and $-y^*$	122
4.5	Labeling neighborhood δ is smaller than the region of maximum amplitude force y^*	123
4.6	Labeling neighborhood δ is larger than the region of maximum amplitude force y^*	123
4.7	Data set composed of 2 classes, blue and red, with 500 instances each one	125
4.8	Classification result for the 1-NN technique	126
4.9	Classification result of the technique based on interacting forces	126
4.10	Example of a 2-dimensional artificial data set used in simulations.	127
4.11	Analysis of the influence of parameters β and δ on accuracy and convergence time for Gaussian artificial data sets (see Fig. 4.10)	128
4.12	Analysis of the data set size and the number of initially labeled objects for 10-dimension Gaussian artificial data sets (see Fig. 4.10)	129
4.13	Analysis of parameters β and δ on the <i>Digit1</i> benchmark data set	130
4.14	Analysis of parameters β and δ on the <i>Digit1</i> benchmark data set	131
5.1	A small network composed of 8 vertices used to illustrate the calculation of the consensus dissimilarity	137
5.2	State evolution of vertices of the network depicted in Fig. 5.1	138

5.3	Consensus dissimilarities between some pairs of neighbor vertices of the network depicted in Fig. 5.1	138
5.4	Networks constructed by using the Alg. 11 from an artificial data set . .	143
5.5	Relation of parameter k to the data set cluster dispersion	147

List of tables

3.1	Metadata of data sets used in simulations.	60
3.2	Classification accuracy (%) followed by standard deviation of the ease of access technique.	63
3.3	Rank of algorithms followed by standard deviation for the results in Table 3.2.	64
3.4	Classification results for the toy example showed in Fig. 3.3 using high-level classification.	74
3.5	Classification accuracy (%) followed by standard deviation for the high-level classification.	77
3.6	Classification accuracy (%) followed by standard deviation for some classification techniques.	84
3.7	Rank of algorithms followed by standard deviation for the results in Table 3.6	85
3.8	Classification accuracy (%) followed by standard deviation for the ETH-80 image collection using high-level classification.	88
3.9	Some representative cases of the ETH-80 classification results.	89
3.10	Classification accuracy average (%) and standard deviation for the ETH-80 images data set using multiple observation sets classification.	97
3.11	Metadata of the simulated data sets for dimensionality reduction.	101
3.12	Classification accuracy (%) using the reduced projected attribute space after dimensionality reduction performed by 3 underlying networks: kAOG, k-NN and ϵ -radius.	102
3.13	Projected low-dimension used for classification after dimensionality reduction for the results in Table 3.12.	102
4.1	Meta-data of the data sets composing the SSL benchmark.	114
4.2	References for the simulated semi-supervised techniques.	116
4.3	Classification error rate (%) and the average rank for semi-supervised techniques. Data sets with 10 labeled points.	117
4.4	Classification error rate (%) and the average rank for semi-supervised techniques. Data sets with 100 labeled points.	118
4.5	Predictive errors (%) on the benchmark data sets provided in (Chapelle <i>et al.</i> , 2006) for the attraction forces technique.	133
4.6	Rank of results for 10 initially labeled instances (see Table 4.5).	134
4.7	Rank of results for 100 initially labeled instances (see Table 4.5).	134

5.1	Clustering accuracy of the proposed consensus technique using different hierarchical clustering methods	145
5.2	Clustering accuracy of the consensus technique calculated by the ARI.	148
5.3	Rank of techniques for the results in Table 5.2.	149

List of Algorithms

1	Calculation of vertex betweenness	29
2	Manifold-based smoothing under constraints	50
3	Network-based classification by random walk ease of access	58
4	High-level network-based data classification	71
5	Classification of multiple observation sets via modularity	93
6	Classification of multiple observation sets via Katz centrality	94
7	k-Associated Optimal Graph	100
8	k-Associated Graph	100
9	Semi-supervised transductive classification via random walk limiting probabilities	112
10	Semi-supervised classification based on interacting forces	119
11	Network construction based on the single-linkage clustering heuristic	142

List of symbols

Notation	Description
A	Network adjacency matrix.
\mathcal{E}	Set of network edges (links).
\mathcal{G}	A graph.
\mathcal{G}_l	Network composed of vertices of class label l .
\mathcal{L}	Set of labels.
\mathcal{N}	A network.
\mathcal{T}	Set containing the τ states with the largest limiting probabilities.
\mathcal{V}	Set of vertices.
\mathcal{V}_l	Network measure vector for all data items of class label l .
V_i	Lyapunov candidate function.
\mathcal{W}	Network connection matrix.
$\hat{\mathcal{W}}$	Modified connection matrix.
\mathcal{W}_{sim}	Biased connection matrix.
\mathcal{W}_{str}	Structural connection matrix.
\mathcal{X}	Data set.
$ \mathcal{X} $	Cardinality of data set \mathcal{X} (number of elements)
$\mathcal{X}^{(l)}$	Data set composed of labeled items.
$\mathcal{X}^{(u)}$	Data set composed of unlabeled items.
\emptyset	Empty set.
B	Penalty connection matrix.
C^{Katz}	Katz index in matrix form.
E	Number of network edges (links).
L	Laplacian matrix.
Q	Network modularity.

Notation	Description
\mathcal{S}	Vector of similarities between an unlabeled instance and labeled instances.
$\mathcal{S}^{(l)}$	Vector of similarities between a labeled instance and unlabeled instances.
V	Number of network vertices.
α	Tuning parameter for convex combination in the high-level classification.
β	Parameter to control force function width.
δ	Labeling neighborhood region.
ϵ	Step size in dynamical pinning control.
γ	Threshold for network construction via single-linkage clustering heuristic.
ϕ	Attraction force amplitude parameter.
ψ	Linear self-feedback weighting parameter in dynamical pinning control.
c	Network clustering coefficient.
c_i	Clustering coefficient of vertex i (v_i).
c_i^{Katz}	Katz index of vertex i (v_i).
e_{ij}	Network edge (link) between vertices i and j .
ϵ	Weighting parameter for the composition of the unlabeled bias links.
k	Number of nearest neighbors.
k_i	Degree of vertex i (v_i).
$\langle k \rangle$	Average network degree.
l	Data label.
m	Number of unlabeled data items.
n	Number of labeled data items.
n_l	Number of items in class l .
\mathbf{p}^∞	Vector of random walk limiting probabilities.
$o_i(\mathbf{p})$	An observation (transformation) of pattern \mathbf{p} .
q	Number of data features (attributes).
r	Network assortativity.
s_i	Similarity between an unlabeled node and a labeled node v_i .
t	Time index.
v	Network vertex (node).
$\mathbf{x}^{(l)}$	Labeled data item.
$\mathbf{x}^{(u)}$	Unlabeled data item.
$\dot{\mathbf{x}}(t)$	Dynamical oscillator.
z	Control gain in dynamical pinning control.

Introduction

The main goal of this thesis is to develop new machine learning techniques by exploring advantages of dynamical processes on complex networks. Therefore, three main research areas are involved: machine learning, complex networks and dynamical processes.

Machine learning

Researches on machine learning techniques and applications have been increasing more and more in diverse areas such as computer science, engineering, medicine, physics, biology etc (Theodoridis and Koutroumbas, 2008). Consequently, there are many different approaches to perform learning tasks. The groups of **supervised** and **unsupervised** learning paradigms are examples of a traditional categorization (Mitchell, 1997). The supervised learning aims to find a rule which predicts the output of a given input data, that is, it tries to find relationships between input-output data pairs in a way that the prediction rule is more accurate as more labeled examples are given. Labeled examples are those for which some information is known such as their classes or groups. On the other hand, the unsupervised learning paradigm seeks underlying structures in a given data set, working specifically with unlabeled instances, that is, data of which classes or groups are unknown. There is also a more recent and intermediate paradigm called **semi-supervised** learning, which is characterized by the usage of the information extracted from just a few labeled data samples, while the majority of the input data is unlabeled (Chapelle *et al.*, 2006).

The supervised learning comprises the construction of a predicting model by using information extracted from a training data set. The constructed model defines decision

borders that are used to classify unlabeled data (Duda *et al.*, 2000). An unlabeled instance is classified depending on its relative position to the decision borders. Due to their importance in various real applications, many classification techniques have been developed, such as Linear Discriminant Analysis (LDA) (Duda *et al.*, 2000), Neural Networks (Haykin, 1998), k -Nearest Neighbors (k -NN) (Tan *et al.*, 2005), Naive-Bayes (Friedman *et al.*, 1997), Support Vector Machines (SVM) (Burges, 1998) and Decision Trees (Alpaydin, 2009).

However, a problem can arise when a supervised technique requires labeled instances that are hard to provide. For instance, if one wants to classify a group of web pages over the Internet accordingly to their areas of interest, such as news, literature, movies, sports etc, it becomes an arduous task to provide many labeled examples because the Internet hosts billions of pages, and the initial categorization of each web page must be performed by a human or expert. In this case, the labeling task becomes expensive and time-consuming (Chapelle *et al.*, 2006).

Consequently, the semi-supervised paradigm arose to alleviate or even overcome this problem. The main idea behind this paradigm is to classify data by using the information extracted from just a few labeled instances and from a large amount of unlabeled data (Chapelle *et al.*, 2006). Many semi-supervised algorithms have been proposed, which are mainly developed from generative models, including the Gaussian mixture model (Shahshahani and Landgrebe, 1994), mixture of experts (Miller and Uyar, 1996) and extensions (Fujino *et al.*, 2005; Nigam *et al.*, 2000a), transductive and semi-supervised support vector machines (Chapelle and Zien, 2005; Collobert *et al.*, 2006), and boosting algorithms (Loeff *et al.*, 2008; Mallapragada *et al.*, 2009). Another important methodology is called co-training (Blum and Mitchell, 1998).

Data clustering is the most prominent branch of unsupervised learning. It consists in dividing an input data set in groups by following a specific criterion: data items within the same group are more similar among them than to data items belonging to other groups. This is an unsupervised learning task because data items are not labeled and the number of groups is usually unknown *a priori* (Mitchell, 1997). Data clustering has been used for the following three main purposes: i) discovering the underlying structure: to obtain insight into data, generate hypotheses, detect anomalies and identify salient features; ii) performing natural classification: to identify the degree of similarity among forms or organisms (phylogenetic relationship); and iii) realizing data compression: as a method for organizing the data and summarizing it through cluster prototypes (Jain, 2010). The practical applications of data clustering includes the characterization of groups of customers based on purchasing patterns, categorization of Web documents, clustering of genes and proteins sharing similar features, image analysis, among many others (Duda *et al.*, 2001). As representative techniques, we can cite k -Means (Jain, 2010; MacQueen, 1967), CLARANS (Ng and Han, 2002), DBSCAN

(Kryszkiewicz and Skonieczny, 2005), CURE (Guha *et al.*, 1998), ROCK (Guha *et al.*, 2000) and Expectation Maximization (Dempster *et al.*, 1977).

Complex networks ¹

In recent years, a tremendous interest has been devoted to the study of statistical and dynamical properties of large-scale graphs with complex structures. This research field, known as complex networks, usually considers large scale graphs with nontrivial connection patterns (Albert and Barabási, 2002; Newman, 2003a). Such graphs have emerged as a unified representation of complex systems in various branches of science (Bornholdt and Schuster, 2003). This is motivated by the fact that complex networks occur commonly in nature and are essential for the infrastructure of a modern society. Examples of such networks include the Internet, the World Wide Web, telecommunication systems, power grids, social networks, traffic networks and biological networks such as neural networks, gene regulatory networks and protein-protein interactions etc (Newman, 2003b).

The first study on large networks was carried out by Erdős and Rényi, who analyzed randomly connected networks rigorously (Erdős and Rényi, 1961). They proposed a model to generate random graphs with V vertices and E edges, called ER graphs, where for large V and fixed average degree $\langle k \rangle$, the degree distribution is well approximated by a Poisson distribution. Nevertheless, real networks are rarely pure random networks. The study of several dynamical processes over real networks has pointed out the existence of shortcuts, that is, bridging links that connect different areas of the networks, thus speeding up the communication among distant nodes, which is called the small-world effect. For this reason, in the seminal paper of Watts and Strogatz (1998), they have proposed to define small-world networks as those networks having both a small value of average shortest path length like random graphs, and a high clustering coefficient like regular lattices. A few months later, Barabási and Albert (1999) discovered that the degree distribution of many complex networks obeys a power law $P(k) \approx k^{-\gamma}$, where k is the number of links or the degree of a randomly chosen node and γ is the scale exponent, henceforth the term scale-free networks.

After those seminal works, complex networks have become an active field in non-linear science and extensive research has been carried out on the following non-exhaustive topics: network growth and self-organization, degree and betweenness distribution, complex network resilience and cascading breakdown, epidemiological process, community structure, and network stability and synchronization (Albert and Barabási, 2002; Boccaletti *et al.*, 2006; Cohen and Havlin, 2010; Newman, 2003b).

¹The words **network** and **graph** are used interchangeably in this thesis by convention.

Dynamical processes on complex networks

The research field about dynamical processes on complex networks is vast. The recent discovery of the relevance of network representations in many areas of science has stimulated the study of models in which node interactions occur in networks whose topologies are more complex than in previously described cases (Barrat *et al.*, 2012). A first motivation for such studies stems from the fundamental interest in understanding how the topology of the interactions change the nature of phase transitions. These phenomena can be understood as the change occurred in the macroscopic behavior (network level) after some modification in a microscopic property at node level (Barrat *et al.*, 2012).

There are two main modeling schemes when dealing with dynamical processes on networks. In the first one, each node of the network is considered as a single individual or element of the system. The second model regards the processes as dynamical entities such as people, information packets, energy or matter flowing through a network of which nodes identify locations where the dynamical entities transit. In both cases, the dynamical description of the system can be achieved by introducing for each node the notion of a corresponding variable characterizing its dynamical state (Barrat *et al.*, 2012).

As an example of dynamical process considering each node as a single element of the system, we can cite the synchronization of coupled oscillators (Boccaletti *et al.*, 2002). When a large number of elements is coupled through interactions in a complex network, various types of synchronization behaviors can occur. The equality of all internal variables is called complete synchronization and it is the most commonly studied synchronization phenomenon (Pecora and Carroll, 1990). When there is a locking of phases while the correlation between amplitudes is weak, we face a phase synchronization, a weaker form of synchronization (Pikovsky and Kurths, 2001). Another kind of synchronization that is harder to be observed is the generalized synchronization where, in the case of two oscillators, the output of one unit is constantly equal to a certain function of the output of the other unit (Barrat *et al.*, 2012; Rulkov *et al.*, 1995). Specific studies about synchronization on networks include time-discrete maps (Dirickx, 2004), synchronization in small-world systems (Barahona and Pecora, 2002), the paradox of heterogeneity in power-law degree distributions (Motter *et al.*, 2002), non-linear coupling with fire and pulse neurons (Timme *et al.*, 2002), synchronization in general network topology (Belykh *et al.*, 2006), and so on.

The second scheme of dynamical processes on networks, which considers dynamical entities flowing through a network, includes diffusion processes like epidemic spreading (Colizza and Vespignani, 2008; Vogels and Birman, 2003), rumor and information propagation (Moreno and Pacheco, 2004; Zanette, 2002), packets traffic analysis

(Cupertino and Zhao, 2011; Guimerà *et al.*, 2002; Zhao *et al.*, 2005a, 2007), cascading failures (Motter and Lai, 2002; Zhao *et al.*, 2004, 2005b), random walks (Burioni and Cassi, 2005; Noh and Rieger, 2004), among others (Barrat *et al.*, 2012).

1.1 Objectives

The goal of this thesis includes the research on machine learning techniques and heuristics based on complex networks and dynamical processes on networks. The investigation efforts are threefold: we are going to explore and formulate applications for supervised, semi-supervised and unsupervised learning paradigms. The specific objectives for each paradigm are described next.

Objectives for network-based supervised learning

1. To develop a new supervised classification technique which takes into account the **ease of access** of unlabeled instances to training classes through an underlying network with the following features:
 - differently from traditional classification heuristics, the proposed scheme uses random walk limiting probabilities to measure the limiting state transitions among training nodes;
 - due to the dynamical property of the technique, both local and global relationships among data instances are taken into account in the classification phase.
2. To develop a new supervised classification technique which combines the **ease of access** heuristic and the network **topological structure** to characterize data classes with the following features:
 - generally speaking, traditional data classification techniques share the same approach: to divide the data space into non-overlapped or slightly overlapped subspaces (classes) according to physical features of the training data, such as distance, similarity or distribution. In contrast, network-based learning techniques allow one to understand classification problems according to both physical features and semantic meanings of the input data;
 - under the hypothesis that data classes can be characterized by the underlying network topological structure, the previous scheme is extended to a quite general framework in such a way that one can put various network measures of interest in the connection matrix of the underlying data network to guide the random walker.

3. To develop network-based techniques applied to the classification of **multiple observation sets** as follows:
 - multiple observations of an object may be produced and stored in a database. For example, in the case of sensor networks the images of an object are captured from different viewpoints for further analysis. A car being tracked via a road camera system or a person having his/her images stored by an internal vision sensors network are examples of objects captured at different time instants or at different angles and geometric transformations;
 - a set of observations consists of different transformations, possibly including rotation, perspectives and projections. Each set belongs to a single pattern, that is, the pattern is considered invariant under such transformations;
 - we pursue the development of efficient network-based methods with the abilities to characterize topological structure of input patterns, to explore these multiple views of invariant patterns to predict and to correctly extract relevant information in classification tasks.
4. To study the usage of different network formation methods into a graph embedding framework to perform supervised **dimensionality reduction** with the following characteristics:
 - some kinds of data, specially images, are often high-dimensional patterns. Dimensionality reduction can enhance processing and also increase classification accuracy;
 - the developed technique maps data into networks and constructs two adjacency matrices to convey information about intra-class components and inter-class penalty connections. The penalty network conveys information about which data samples (class components) should not be close together in the reduced feature space.

Objectives for network-based semi-supervised learning

1. To extend the previous **ease of access** heuristic to the semi-supervised setting as follows:
 - instead of constructing a training network from the labeled instances, the set of unlabeled instances is used to create a network, and the link weight composition takes into account the information provided by the labeled instances;

- each label is propagated through a network of unlabeled instances via a bi-ased random walk. The random walk process measures the label propa- gation from labeled vertices to the remainder unlabeled vertices of the net- work.
2. To develop a nature-inspired semi-supervised classification technique based on **attraction forces** as follows:
 - data instances are represented as points in a q -dimensional space and the movement of data points is modeled as a dynamical system. As the system runs, data items with the same label cooperate with each other and data items with different labels compete among them to attract unlabeled points by applying a specific force function;
 - the use of attraction forces between labeled and unlabeled instances can pro- vide a model for semi-supervised learning that fits well into the smoothness and cluster assumptions. Labeled instances are considered as fixed attrac- tion points that apply attraction forces on unlabeled instances. In turn, the unlabeled instances are expected to move towards the resultant force direc- tion and, eventually, to converge to an attraction point.

Objectives for network-based unsupervised learning:

1. To develop a data clustering technique based on **synchronization and pinning control** of networked dynamical oscillators with the following features:
 - in order to overcome some limitations of classical data clustering methods such as the assumption of clusters with hyper-spherical shapes and similar sizes, we study a new network-based data clustering method;
 - a dissimilarity measure is computed via a dynamic system in which ver- tices are expected to reach a consensus state regarding a reference trajectory forced into the system via pinning control. The resultant dissimilarity pro- vides a set of features for the data items allowing the detection of clusters with different shapes and sizes.

1.2 Motivations

Networks are powerful tools for complex systems modeling and for data represen- tation and analysis due to the following salient features. i) Networks are ubiquitous in nature and everyday life. Many are the networked systems such as the Internet,

World Wide Web, telecommunication networks, transportation networks, biological networks and social networks. Nevertheless, many other types of data sets can be transformed into networks. For example, a vector-based data set can be transformed into a network by simply connecting to each node its k most similar neighbors. ii) Network representation unifies the structure, dynamics and functions of the system it represents. Besides describing the interaction among nodes (structure) and the evolution of such interactions (dynamics), it also reveals how the pair "structure + dynamics" affects the overall function of the network (Boccaletti *et al.*, 2006). For instance, it is well known that there is a strong correlation between the structure of protein-protein interaction networks and the protein functions (Palla *et al.*, 2005; Spirin and Mirny, 2003). iii) One of the main motivations of graph theory research is its ability to describe topological structures of complex systems. Such representation emphasizes the physical distances among nodes and captures from local to global relations among data.

In the machine learning domain, the topological structure is quite useful to detect diverse cluster or class shapes via a data clustering or classification algorithm, respectively. As a consequence, the application of network-based methods in learning tasks has been increasing over the past years and has become a active research area with a myriad of applications such as semi-supervised learning (Breve *et al.*, 2012; Chapelle *et al.*, 2006; Nguyen and Mamitsuka, 2011; Silva and Zhao, 2012a), clustering (Karypis *et al.*, 1999; Schaeffer, 2007; Silva and Zhao, 2012b; Silva *et al.*, 2013), regression (Ni *et al.*, 2012), feature selection (Bunke and Riesen, 2011), dimensionality reduction (Riesen and Bunke, 2009), among others.

For instance, traditional clustering techniques are very efficient in some cases, although they may be not suitable for some other cases. As an example where such algorithms are not appropriate is the clustering of data sets that contains groups of different spatial shapes, sizes and densities, as those methods generally assume that the clusters are hyper-spherical and have similar sizes. On the other hand, network-based data clustering methods are able to detect groups with arbitrary shapes because networks are powerful tools to represent topological relations among objects. These methods usually consist of two stages: construction of a network from the original vector-based data set, and partition of the network into sub-networks, each one representing a data cluster.

In a network, the concept of cluster is regarded as a densely connected group of vertices, while the connections among different clusters are sparse. A representative method for data clustering based on networks is called Chameleon (Karypis *et al.*, 1999), which uses the concept of k -NN on a sparse network representing the data set. The algorithm searches for the topological structure of the input data, and so it is able to identify data clusters with different shapes. However, the network formation by using k -NN presents two main drawbacks: 1) the resulting network is not necessarily

connected, and 2) the resulting network may be dense. In these cases, it is difficult to correctly divide the network into meaningful sub-networks.

There are other methods for detecting clusters and/or communities in complex networks that have been showing good results. A well-known method is based on an iterative removal of edges that present high value of a measure called *betweenness* (Freeman, 1977; Newman and Girvan, 2004), resulting in a divisive hierarchical tree of communities. Also, the method of optimal modularity (Newman, 2006) considers the community structure as a statistical arrangement of edges, which can be quantified by using a measure known as modularity (Newman, 2004). Another method of community detection is based on the concept of collective intelligence (Oliveira and Zhao, 2008). In this method, all vertices are randomly arranged in a circle, so that the angles of each vertex are gradually updated according to the angles of their neighbors. At end of the process, the vertices reach a stationary state in which vertices belonging to the same community lie grouped together. In (Quiles *et al.*, 2008), the authors proposed a method based on particle competition dynamical process in which particles move through the vertices and compete among themselves to dominate as many vertices as possible. Eventually, each particle will dominate a single community. In (Silva *et al.*, 2013), the authors applied the same concept of agents competition to cluster handwritten images of alphanumeric digits. Another method of community detection (Zhou, 2003a,b) uses a distance measure of complex networks based on the random walk of a Brownian particle through the network. Methods that use the idea of synchronization of coupled oscillators in a network (Zhao *et al.*, 2008a) consider that each group of vertices that synchronize in similar times represent a data cluster. When the synchronization is achieved via pinning control (Li *et al.*, 2008), the communities are synchronized to a common state either in the phase space or in time by controlling some vertices of the network.

The motivations for the research in the supervised learning paradigm is mainly due to the possibility of different classification heuristics. Traditional classification techniques divide the data space according to physical features of the training data (similarity, distance, or distribution). They divide the data space into subspaces, each one representing a data class. These subspaces are not overlapped in the case of crisp classification, but they can be slightly overlapped in the case of fuzzy classification. In either way, strong twisting or largely overlapped subspaces are not permitted. In this way, many intrinsic and semantic relations among data items are ignored as, for example, topological structures. On the other hand, a relevant advantage when using networks in learning tasks is that it can perform quite different classification heuristics. For example, it can take into account topological structures and pattern formation.

Furthermore, the number of network-based supervised techniques reported so far is still small (Bertini *et al.*, 2011). At a first glance, several network-based semi-

unsupervised methods, which are more expressive in numbers than the supervised ones, such as those presented in (Belkin *et al.*, 2006; Sindhwani *et al.*, 2005), could be converted to the supervised setting if a reasonable number of labeled instances was provided. However, these methods consider 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 *et al.*, 2011).

Interestingly, supervised dimensionality reduction can also be performed by using a network embedding framework (Yan *et al.*, 2007). The purpose of network embedding is to represent each node (data sample) of a network as a low-dimensional vector that preserves similarities between the node pairs, where similarity is measured by a network similarity matrix that characterizes certain statistical or geometric properties of the data set. The usage of network embedding for dimensionality reduction can overcome some limitations of the LDA technique such as the number of available projection directions lower than the number of classes, and the assumption that data is approximately Gaussian distributed (Yan *et al.*, 2007).

In the real world, there is a necessity of capturing and storing in a database different observations of a given pattern. For example, in a sensor network, signals of an object are captured via sensors positioned at different places, resulting in different observations of the same object. Also, a car being tracked via a road camera system or a person having his/her images stored by an internal vision sensors network are examples of objects captured at different time instants or by different angles and geometric transformations (Zhao *et al.*, 2008b). Thus, efficient methods must be developed to exploit these multiple views of invariant patterns to predict and correctly extract relevant information.

In this scenario, a network-based scheme permits the detection of subnetworks which represent each group of observations, and the classification process takes into account the overall link strength between two subnetworks. For example, the topology of a network may dictate some intrinsic importance for different vertices and this importance can be measured by using different measures (Cohen and Havlin, 2010). A centrality measurement called Katz centrality is an example (Katz, 1953). This metric is capable of accounting the importance of a vertex considering the paths of different lengths starting at it to the other vertices over the network. Thus, regarding each vertex of the network as a stored pattern or a pattern transformation, it is possible to measure the relation degree between an original pattern and its transformations: the more strong is the centrality of a transformation in respect to the original patterns, the more is the transformation related to that pattern's class.

In summary, network-based techniques presents some salient advantages over

other traditional techniques for all the three categories of machine learning.

1.3 Organization of this thesis

The remainder of this thesis is organized as follows. In Chapter 2, we review the theoretical basis for the developments presented in the subsequent chapters. Namely, we review concepts related to machine learning, complex networks and dynamical processes on networks, as well as some relevant network-based techniques applied to machine learning that use concepts of dynamical processes. Chapter 3 starts introducing the research results. There, four new network-based supervised techniques are presented: a classification technique based on the concept of ease of access to data classes by using random walk, and its extension which considers the network topological structure in a general classification framework called high-level classification; a technique for classification of multiple observations sets based on network measures; and an application of a specific network construction method into an embedding dimensionality reduction framework. In Chapter 4, we present an extension of the supervised random walk ease of access to a semi-supervised transductive setting, and a semi-supervised technique based on interacting forces, in which data samples are modeled as points in a networked interaction scheme. The results concerning unsupervised learning are presented in Chapter 5. Specifically, we introduce a data clustering network-based technique which uses the concept of pinning control of dynamical processes. The conclusions are shown in Chapter 6, as well as some directions for future studies and a list of the published articles during the doctorate period.

Review of essential concepts

This chapter introduces the fundamental concepts related to the developments presented in this thesis. Three major areas are introduced: machine learning, complex networks and dynamical processes on networks. Machine learning deals with the essential ideas and heuristics to develop algorithms capable of extracting useful knowledge from data sets. It is commonly divided into three sub-areas: supervised learning, when there is information about data classes involved in the learning process; semi-supervised learning, when there is just a little bit of relevant information about data classes in comparison to the whole data set; and unsupervised learning, when there is no *a priori* knowledge about the data classes we are dealing with.

Complex networks is a research field dedicated to the study of systems represented by large scale graphs. The studied systems are called complex due to their non-trivial connection pattern and evolution of the network structure. Many real-world systems are complex networks, such as biological networks, telecommunication networks, Internet pages and social networks. The main topics include the characterization of topological structures of a given family of networks, the study of dynamical processes through the nodes and edges, how nodes and links relate to each other and what is their role or function in a given topological structure.

The third section reviews some relevant studies concerning dynamical processes on networks. It presents a detailed introduction about random walk processes and the related touristic walk. The dynamical processes known as node synchronization and consensus are also introduced. These concepts are the basis for the techniques presented in the last section of this chapter, where label propagation and supervised network construction are also described. These essential ideas on dynamical processes and network-based techniques form the theoretical foundations for the developments

described in the next chapters.

2.1 Machine learning

The learning skill can be naturally found and perceived in human beings. We are born with this amazing ability which is further developed by accumulated experience during our lives. Machine learning is a research area of computer science that seeks to develop methods and techniques capable of learning with experience (Bishop, 2007). By representing data from diverse domains in a computer environment, machine learning techniques can generate models capable of organizing the intrinsic knowledge found in the data, or even imitating the behavior of a specialist in the considered domain (Duda *et al.*, 2000). The goal is to develop techniques which enable a computer, or a machine, to learn from the data under analysis.

Generally speaking, the machine learning techniques are classified into one of two distinct paradigms: supervised learning and unsupervised learning (Mitchell, 1997). In supervised learning, the goal is to infer data labels by using some labeled data samples from *a priori* known classes, that is, the learning process aims at constructing a mapping function by using the information provided by training labeled data. When labels are discrete, the problem is called classification and, when labels are continuous, it is called regression (Bishop, 2007). In the case of unsupervised learning, the main task is to cluster data by following a similarity criterion where the whole process is guided solely by the data, given that there is no prior knowledge about data classes (Bishop, 2007; Mitchell, 1997).

The supervised learning techniques can be further categorized into inductive and transductive. In inductive learning, the goal is to learn a function that makes predictions on the whole data space. On the other hand, transduction asks for less - it only concerns itself with predicting the values of the function at the test points of interest, that is, not on the whole data space. This can be viewed as an easier problem, since an inductive solution implies a transductive one - by evaluating the function at the given test points - but not vice versa (Chapelle *et al.*, 2006).

Recently, another learning paradigm called semi-supervised learning has received attention in the research community (Chapelle *et al.*, 2006). Techniques belonging to this learning paradigm try to overcome the problem arisen when a supervised technique requires labeled instances that are hard to provide. For instance, if one wants to classify a group of web pages over the Internet according to their areas of interest such as news, literature, movies, sports etc, it becomes arduous to provide many labeled examples because the Internet hosts billions of pages, and the initial categorization of each web page is usually performed by an expert human. In this case, the labeling task becomes expensive and time-consuming. Thus, the main idea behind the

semi-supervised learning is to perform classification tasks by using both: a few labeled instances and the information provided by a large amount of unlabeled instances. This approach can provide higher accuracies along with less human effort by exploiting the unlabeled massive group of data (Seeger, 2002; Zhu and Goldberg, 2009).

In the following, we provide a brief review of each of the learning paradigms and some of their most relevant techniques.

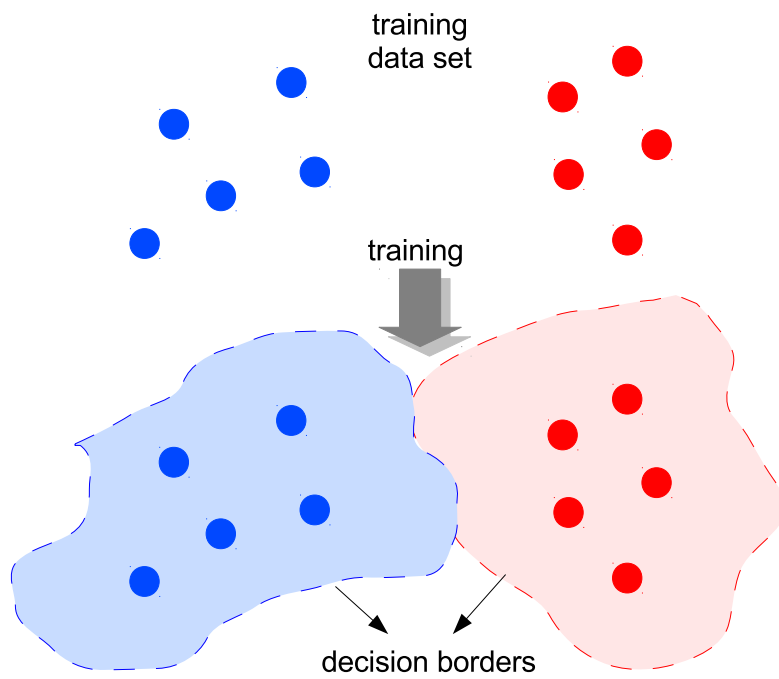
2.1.1 Supervised learning

Formally, the supervised learning paradigm can be posed as follows (Bishop, 2007; Mitchell, 1997). The problem requires a labeled data set, $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, where each training instance has a single assigned label $l \in \mathcal{L}$, where \mathcal{L} is the set of labels and is described by q attributes $\mathbf{x}_i^{(l)} = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$. The available labels, or classes, are disjoint. The goal is to learn a classifier capable of mapping unlabeled instances $\mathbf{x}^{(u)}$ to their related labels l , that is, $\mathbf{x}^{(u)} \mapsto l$. The phase in which the classifier is induced is called training phase. Oftentimes, the constructed classifier is tested by using an unlabeled test data set $\mathcal{X}^{(u)} = \{\mathbf{x}_j^{(u)}, j = 1, \dots, m\}$, for which labels are known but are not provided. Thus, the second step is called test phase and it aims at enabling a measure for the accuracy rate of the trained classifier. Furthermore, to avoid biased learning, $\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$. Figure 2.1 depicts both phases. In Fig. 2.1a, the labeled data is used to infer decision borders and, in Fig. 2.1b, the trained classifier labels unlabeled data.

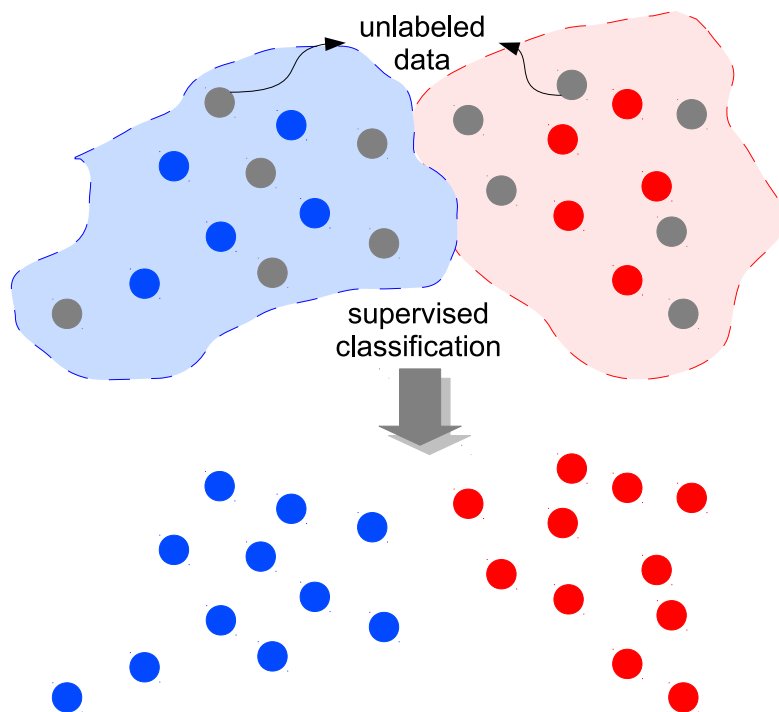
Some assumptions concerning the training and the test sets must be fulfilled in order to learn a classifier with good generalization performance. Two important examples are: i) **Representative training set** - the training set must be a representative sample concerning the distribution or population that generated it. Since the classifier induction is based upon the training set, if it is not a representative sample of the data distribution, it is likely to mislead classification to predict in accordance with a different distribution; ii) **Unbiased test set** - the test set should be unbiased towards the training set in order to achieve a valid estimation, but it must be sampled from the same distribution that generates 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 inferring unseen examples of the same distribution. This assumption is often violated to a certain degree in practice, and strong violations results in poor classification rate accuracies (Bishop, 2007; Mitchell, 1997).

As examples of traditional supervised classification techniques, we list the following:

- **Instance based learning:** it comprises the nearest neighbor decision rule (k -NN), which assigns a label from a set of nearest neighbors to an unlabeled sample



(a) Training phase.



(b) Test phase.

Figure 2.1: Illustration of training and test phases in supervised classification. (a) The training phase is responsible for using the labeled data to learn decision borders. (b) The trained classifier labels unlabeled data.

point. This rule is independent of the underlying joint distribution on the sample points and their classifications and, hence, the probability of error R of such a rule must be at least as great as the Bayes probability of error R^* - the minimum probability of error over all decision rules taking underlying probability structure into account (Cover and Hart, 1967). This classification rule is strongly based on the distance function used to calculate similarity between data instances;

- **Decision trees:** a decision tree is a simple recursive structure for expressing a sequential classification process in which a case, described by a set of attributes, is assigned to one of a disjoint set of classes. Each leaf of the tree denotes a class. An interior node denotes a test on one or more attributes with a subsidiary decision tree for each possible outcome of the test. To classify a case, we start at the root of the tree. If this is a leaf, the case is assigned to the nominated class; if it is a test, the outcome for this case is determined and the process continued with the subsidiary tree appropriate to that outcome (Quinlan, 1986, 1992);
- **Production rules:** these rules are often expressed as a group of IF-THEN clauses. 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 part (Quinlan, 1987). Production rules of decision trees are vehicles for representing knowledge in expert systems and improving classification performance by eliminating tests in the decision tree attributable to peculiarities of the training set and by making it possible to combine different decision trees for the same test (Quinlan, 1987);
- **Probabilistic graphical models (Bayesian networks):** a Bayesian network is a acyclic graph in which nodes correspond to variables and edges denote a probabilistic dependency between two connected nodes. The graph needs to be directed to describe a directionality between two node variables. These networks represent the joint probability distribution over a set of random variables taking into account conditional independencies among them. A Bayesian network consisting of a class variable and feature variables is applicable to the classification task because the conditional probability distribution can be calculated from the joint probability distribution (Koller and Friedman, 2009). The class labels are predicted by using the maximum *a posteriori* estimate obtained by the Bayes rule. These networks require parameter learning (Pernkopf *et al.*, 2012; Roos *et al.*, 2005). A special case of Bayesian networks is when there are strong (naive) independence assumptions on the predictive variables. In this case, the classifier is best known as naive Bayes (Neapolitan, 2003);
- **Artificial neural networks:** neural networks are a connectionistic approach for

supervised learning. Each processing unit, a neuron (Rosenblatt, 1957), is connected to other neurons via weighted links (synapses). As the information flows from the input to the output of the network, the synaptic weights changes in order to adapt the connections to learn an input-output mapping. A very popular training algorithm is the backpropagation, a generalization of the gradient descent learning rule. In a multi-layered neural network, it acts by propagating errors from the output layer back to the input layer, so the synaptic weights can be updated in order to decrease the error. Important decisions when projecting a artificial neural network include the number of layers, the network structure, the neurons threshold, and the learning algorithm (Haykin, 2008);

- **Statistical learning theory:** let $X \in \mathbb{R}^q$ denote a real valued random output variable with joint distribution $Pr(X, Y)$. The goal is to find a function $f(X)$ for predicting Y given values of the input X . This theory requires a loss function $L(Y, f(X))$ for penalizing errors in prediction, and by far the most common and convenient is squared error loss: $L(Y, f(X)) = (Y - f(X))^2$ (Hastie *et al.*, 2009). As classical statistical learning techniques, we can cite Support Vector Machines (SVM) and Linear Discriminant Analysis (LDA). SVM techniques seek for an optimal separating hyperplane, where the margin between two different classes is maximal. The solution is based only on those data points, the support vectors, that are at this margin. The linear SVM can be extended to a nonlinear one when the problem is transformed into a feature space using a set of nonlinear basis functions. In this feature space - which can have a high dimension - the data points can be separated linearly (Vapnik, 1998). LDA uses the class label information of the input data samples. LDA finds a projection matrix that maximizes the trace of the between-class scatter matrix and minimizes the trace of the within-class scatter matrix in the projected subspace simultaneously (Fukunaga, 1991).

2.1.2 Semi-supervised learning

Somewhat similarly to the supervised learning paradigm, the semi-supervised paradigm can be stated as follows (Chapelle *et al.*, 2006). The problem requires a labeled data set, $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, where each training instance has a single assigned label $l \in \mathcal{L}$, and an unlabeled data set, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, in which instances have no associated labels. Data samples are described by q attributes: $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$. The available labels, or classes, are disjoint. The goal is to learn a classifier capable of mapping unlabeled instances $\mathbf{x}_i^{(u)}$ to their related labels l , that is, $\mathbf{x}_i^{(u)} \mapsto l$. To characterize a semi-supervised task and to differ it from the supervised paradigm, the number of labeled samples is usually much smaller than the number of unlabeled samples, that is, $n \ll m$. Figure 2.2 depicts an example of a semi-supervised

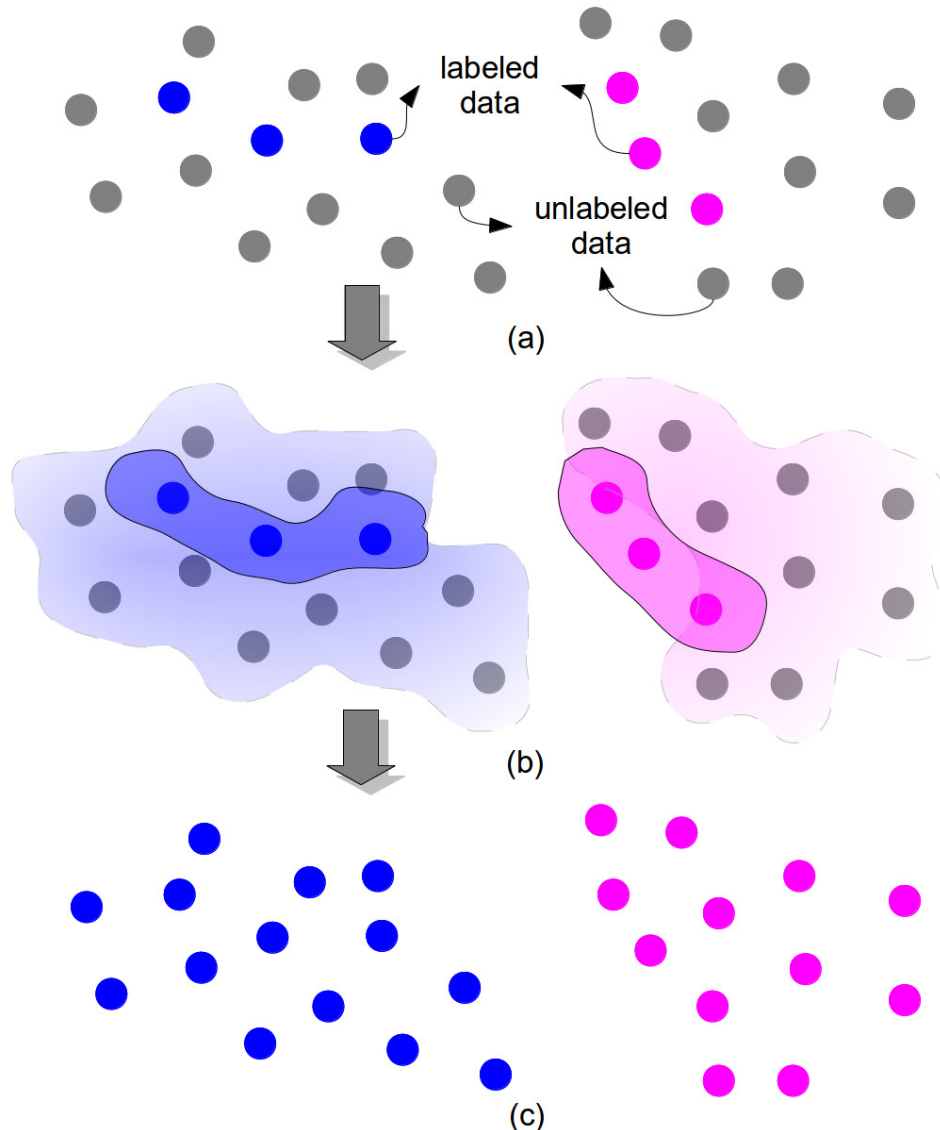


Figure 2.2: Illustration of semi-supervised transductive classification. (a) Labeled and unlabeled data. (b) Regions of influence showing that all data, labeled and unlabeled, are taken into account in the classification process. (c) Classification result.

transductive classification.

As it is mentioned before, in the semi-supervised setting, all data, including labeled and unlabeled, are used either in the training phase (in case of inductive classification) or in the classification phase (in case of transductive classification) (Chapelle *et al.*, 2006). Due to this peculiar characteristic, some assumptions concerning the data distribution must be taken into account in order to conduct the learning process (Chapelle *et al.*, 2006). These assumptions can be stated as follows:

- *Manifold assumption:* the high-dimensional data lies on a low-dimensional manifold whose properties ensure more accurate density estimation and more appropriate similarity measures. This assumption is often used when dealing with the curse of dimensionality that occurs with high-dimensional spaces. In these cases,

as the dimension increases, the data space volume increases exponentially and distances between data samples become large and similar to each other, hampering the classification results (Duda *et al.*, 2000);

- *Smoothness assumption*: if two points are close to each other in a high-density region, then their correspondent labels should be close to each other as well. In other words, this assumption states that the variations of data labels in high-density regions should be smooth;
- *Cluster assumption*: if two points are in the same cluster, then they are likely to belong to the same class (or, in other words, to have the same label). This assumption complements the smoothness assumption when considering clusters as well-defined or high-density regions.

As examples of the categories in which the semi-supervised techniques are commonly grouped (Chapelle *et al.*, 2006), we list the following:

- **Generative models**: the inference via generative models involves the estimation of a conditional density. The Expectation Maximization (EM) technique is the most known technique pertaining to this approach (Nigam *et al.*, 2000b). Besides that, a myriad of techniques proposed so far in the literature can be encountered in Alpaydin (2009); Chapelle *et al.* (2006); Gärtner (2008); Zhu and Goldberg (2009);
- **Cluster-and-label models**: the label 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 label inference is based on the development of decision boundary functions that are created as far as possible from the high-density regions. Undoubtedly, the most known method of this approach is the Transductive SVM (TSVM) (Cortes and Vapnik, 1995; Vapnik, 1998). More related techniques can be found in Alpaydin (2009); Chapelle *et al.* (2006); Cortes and Vapnik (1995); Zhu and Goldberg (2009);
- **Self-training**: in self-training, a classifier is first trained with the small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically, the most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure is repeated. In this way, the classifier uses its own predictions to teach itself. Self-training is a wrapper algorithm, and is hard to analyze in general (Zhu, 2008).

- **Co-training:** it assumes that (i) features can be split into two sets; (ii) each sub-feature set is sufficient to train a good classifier; and (iii) the two sets are conditionally independent given the class. Initially, two separate classifiers are trained with the labeled data on the two sub-feature sets respectively. Each classifier then classifies the unlabeled data, and "teaches" the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats (Blum and Mitchell, 1998; Mitchell, 1999).
- **Graph-based methods:** these methods define a graph where the nodes are labeled and unlabeled samples in the dataset, and edges (may be weighted) reflect the similarity amongst data samples. These methods usually assume label smoothness over the graph. Graph methods are nonparametric, discriminative, and transductive in nature (Zhu, 2008).

In Zhu (2008), the author addresses the issue of which semi-supervised method one should use. Ideally, one should use a method whose assumptions fit the problem structure. In view of that, Zhu (2008) proposes some questions in form of a checklist in order to help finding the best method for the data at hand:

- if the classes produce well-clustered data, EM with generative mixture models may be a good choice;
- if the features naturally split into two sets, co-training may be appropriate;
- if two points with similar features tend to be in the same class, graph-based methods can be used;
- if SVM is already being used, TSVM is a natural extension;
- if the existing supervised method is hard to modify, self-training is a practical wrapper method.

2.1.3 Unsupervised learning

In contrast to the previously introduced supervised and semi-supervised learning paradigms, the unsupervised learning does not use any label information in its original form (Mitchell, 1997). Given an unlabeled data set, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, in which instances have no associated labels and are described by q attributes, $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$, the objective is to find hidden structures within the unlabeled data set. Typically, it is assumed that the points are independently and identically distributed in accordance with a common distribution. As there is no information about

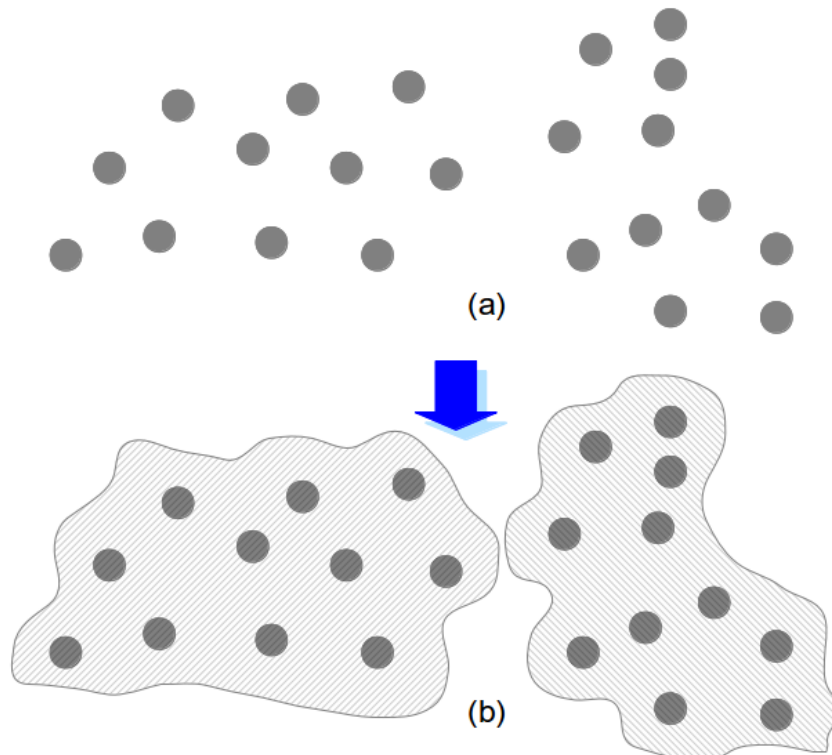


Figure 2.3: Illustration of data clustering process. Unlabeled data set in (a) is clustered into two groups in (b).

labels or classes, an unsupervised technique must be guided solely by the intrinsic information of the data sets to infer useful knowledge (Bishop, 2007; Mitchell, 1997). As a consequence, there is no training phase for these kind of algorithms.

The most known application of unsupervised learning is data clustering. Clustering is ubiquitously used in data mining as a method for discovering novel and actionable subsets within a set of data. Figure 2.3 shows a simple clustering example. The process of clustering is important since, being completely unsupervised, it allows the addition of structure to previously unstructured items such as free-form text documents (Davidson and Basu, 2007). However, the result of a clustering process depends on the assumptions made about the data (Bishop, 2007; Gan, 2007).

Data clustering techniques are usually grouped into two distinct categories:

- **Hierarchical clustering:** this kind of clustering finds successive clusters by using previously established ones and can be divided into agglomerative ("bottom-up") and divisive ("top-down"). Agglomerative clustering starts with the same number of clusters as is the number of data samples, that is, each data sample is a single cluster, and proceeds by grouping clusters to each other to form larger groups. On the other hand, the divisive clustering starts with a large cluster containing all data and proceeds by dividing it into successively smaller clusters. Some of the well-known agglomerative techniques are single-linkage, complete-linkage, average-linkage and Ward methods (Alpaydin, 2009; Mitchell, 1997). Basically,

these techniques differ in the way distances among clusters are calculated. For example, single-linkage considers the distance between two clusters C_1 and C_2 as the minimum distance between x and y , being $x \in C_1$ and $y \in C_2$. A drawback of this method is the so-called chaining phenomenon, which refers to the gradual growth of a cluster as one element at a time gets added to it. In contrast, complete-linkage considers the distance between C_1 and C_2 as the maximum distance between x and y . It tends to find compact clusters of approximately equal diameters (Alpaydin, 2009).

- **Partitional clustering:** given a data set $\mathcal{X}^{(u)}$, the typical aim of partitional clustering is to form a k -block set partition Π_k of the data at once. The k-Means method is one of the most studied partitional clustering technique (Alpaydin, 2009; Mitchell, 1997). Basically, given an initial set of k-Means, it works by alternating between two steps: 1) assign each observation to the cluster whose mean yields the least within-cluster sum of squares, and 2) calculate the new means to be the centroids of the observations in the new clusters. These two steps are performed until the algorithm converges to k distinct clusters. Some of its drawback are the strong dependence of the initial conditions and the bias to find circular-shaped clusters. Trying to overcome these disadvantages, several derivations have been developed by the research community such as k-Medoids and Fuzzy c-Means (Alpaydin, 2009; Jain *et al.*, 1999).

Another classical and very useful example of unsupervised techniques is the Principal Component Analysis (PCA) (Jolliffe, 2002). When data lies in a high-dimensional space, problems can arise when computing similarities among data often due to the "curse of dimensionality" (Duda *et al.*, 2000). PCA is an orthogonal transformation that represent data by using the so-called principal components. Usually, a small number of principal components is sufficient to account for most of the structure in the data. It maximizes the mutual information between the original high-dimensional Gaussian distributed data features and the projected low-dimensional features. In the unsupervised setting, PCA does not use the class label information of the input data. In the supervised setting, data instances are marked with label information that guides the formation of the low-dimensional space. The labels often take discrete class values, indicating which data points have to be grouped together (same class) or set far apart from the other (different classes) in the embedded space (Jolliffe, 2002).

2.2 Complex networks

Graphs are used in order to study and mathematically describe the concepts of complex networks. Graphs represent the topological properties of a network by using

the representation with vertices and links. In a social network, for example, the vertices represent individuals and links represent the relationship that may exist between each pair of individuals. This concise and uniform representation allows the same tools and mathematical methods to be applied equally on both very simple systems and systems that exhibit high complexity (Cohen and Havlin, 2010).

The study of graph theory began in the eighteenth century with the works of Euler while he studied the famous problem of the bridges of Königsberg (now Kaliningrad, Russia). The problem posed the question: could one go through all the seven city bridges without going through a bridge more than once? Euler understood that the major factor was the underlying network topological structure, and thus the problem could be simplified into a study of graph routes in which the vertices represent parts of the city and the links represent bridges between them. The mathematician solved the problem concluding that such path exists if each vertex has an even number of links connected to it, except possibly the first and last vertex visited, which does not occur in the Königsberg graph. Nowadays, such a path is known as the Eulerian path (Lovász *et al.*, 2003).

Still receiving some attention from a branch of mathematicians, the study of graphs showed no substantial advances until the 60s when two mathematicians, Paul Erdős and Alfred Rényi, introduced the theory of random graphs. Basically, their idea was to combine previously developed graph concepts with tools of probability theory, aiming at enable the study of a family of graphs instead of specific samples (Barabási, 2003). Two sets of extensively studied graphs are: $\mathcal{G}_{V,E}$, the set of all graphs with V vertices and E edges, and $\mathcal{G}_{n,p}$, the set of graphs with V vertices in which each possible edge between two vertices exists with probability p . These two sets of graphs are similar if $E = \binom{V}{2}p$ and p is not too close to 0 or 1 (Bolobás, 1985). A well-known result is that they follow a Poisson distribution.

Some decades after, at the end of the 20th century, the advances of computational power and the consequent possibility to analyze large data sets attracted the attention of researchers to real world network structures that would not be properly described by the Erdős and Rényi graph theory. Such networks had link distributions that followed nonuniform probability distributions, so far the only ones considered in the graph theory. The discoveries led to further studies and to the emergence of a new research field called complex networks (Barabási, 2003). These networks encompass new and different topologies such as small-world, scale-free and modular, which best represent the networks found in the real world. Additionally, a large and growing number of measures has been developed to describe them.

The study of several dynamical processes over real networks has pointed out the existence of shortcuts, that is, bridging links that connect different areas of the networks, thus speeding up the communication among distant nodes, which is called the

small-world effect. This feature is found, for example, in social networks, where, virtually, all people of the world can be reached by a small chain of shared contacts (Watts, 2003; Watts and Strogatz, 1998). At the same time, the scale-free networks emerged in a study conducted by Barabási and Albert (Barabási and Albert, 1999), who noted some networks have a small number of vertices with high degree, while the majority of the vertices has very low degrees. Such networks exhibit a degree distribution known as scale-free which obeys the power-law distribution. Furthermore, there are also the modular networks. Generally, modular structures reveal similarities among vertices belonging to the same group accordingly to some criterion. In social networks, for example, the presence of communities can reveal groups of individuals with similar interests, friendships, professional relationships etc. Therefore, the identification of communities in complex networks is essentially a clustering task in a networked data.

In the next sections, some of the most essential concepts and measures of graph theory and complex networks relevant to this work are formalized.

2.2.1 Basics from graph theory

Some of the most basic and key concepts from graph theory are reviewed in this section. These concepts are important in the study of complex networks since such networks are modeled with the same formulation of graph theory, that is, by using the representation of nodes and edges (or vertices and links). For this reason, we use the terms graph and network interchangeably in this thesis. In several references, such as (Diestel, 2006) and (Lovász *et al.*, 2003), deeper discussions about graphs can be found.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is composed of a set of vertices $\mathcal{V} = \{v_1, \dots, v_n\}$ and a set of edges \mathcal{E} . In a **directed graph**, also called a digraph, every edge has a direction, that is, a source and a destination. An edge with origin in vertex $v_i \in \mathcal{V}$ and destination in vertex $v_j \in \mathcal{V}$ is represented by an ordered pair (v_i, v_j) . In **non-directed graphs**, the edges have no direction and an edge that connects v_i to v_j is represented by $e_{ij} = (v_i, v_j) = (v_j, v_i)$. Graphs that present both directed and undirected edges are called mixed graphs. It may happen that the set of edges \mathcal{E} contains multiple instances of the same edge, that is, more than one edge connecting vertices v_i to v_j . In this case, \mathcal{E} is considered a multiset. If an edge occurs more than once in \mathcal{E} , copies of the edge are called parallel edges. Graphs with parallel edges are called multigraphs. When every edge occurs only once in the set \mathcal{E} , the graph is simple.

The edges $e_{ij} \in \mathcal{E}$ can be associated to **weights**. Such weights may be represented by a function $f : \mathcal{E} \rightarrow \mathbb{R}$ that assigns to each edge e_{ij} a weight $f(e_{ij})$. Depending on the context, these weights describe different properties such as time or distance costs, communication skills, interaction strength, similarity, among others. A non-weighted graph is equivalent to a graph with unit weight, $f(e_{ij}) = 1, \forall e_{ij} \in \mathcal{E}$.

A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is a **subgraph** of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. In this case, \mathcal{G} is called a **supergraph** of \mathcal{G}' . An edge induced subgraph corresponds to a graph \mathcal{G}' formed by a subset of edges of the graph \mathcal{G} together with the vertices connected to those edges. Conversely, a vertex induced subgraph, or simply a induced subgraph, refers to a graph \mathcal{G}' formed by a subset of vertices along with their connected edges, of which connected vertices belong to that subset.

Two vertices v_i and v_j are **adjacent** or **neighbors** if there is an edge e_{ij} , $f(e_{ij}) \neq 0$, connecting them. Two edges $e_1 = e_2$ are adjacent if they connect to a common vertex. If every vertex in \mathcal{G} is adjacent to all other vertices, then the graph \mathcal{G} is called complete. An induced complete subgraph is called a click. The set of neighbors of v_i is denoted by Λ_{v_i} , and λ_{v_i} is its number of neighbors.

In an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the **degree** of a vertex v_i , denoted by k_i , is the number of edges in \mathcal{E} that connects to v_i . If \mathcal{G} is a multigraph, parallel edges are counted according to their multiplicity in the set \mathcal{E} . A vertex v_i with $k_i = 0$ is said to be isolated. Being $V = |\mathcal{V}|$ the number of vertices, the average degree of a graph \mathcal{G} quantifies the overall local measures k_i and it is calculated as follows:

$$\langle k \rangle = \frac{1}{V} \sum_{i=1}^V k_i. \quad (2.1)$$

When the degrees of the vertices are added up, each vertex is counted two times so that the number of edges, $E = |\mathcal{E}|$, can be calculated by:

$$E = \frac{1}{2} \sum_{i=1}^V k_i = \frac{1}{2} \langle k \rangle V,$$

where the second equality is based on Eq. 2.1.

The density of a graph is defined as the ratio between the number of edges and the total number of possible edges. In an undirected graph, this ratio is given by:

$$\delta(\mathcal{G}) = \frac{2E}{V(V-1)}.$$

A **route** starting from a vertex v_0 and ending at a vertex v_k corresponds to a sequence $\langle v_0, e_1, v_1, e_2, v_2, \dots, v_{k-1}, e_k, v_k \rangle$ composed of vertices and edges. The path length is defined by the number of covered edges. A route is called a path if $e_i = e_j$ for $i = j$, and a path is simple if $v_i = v_j$ for $i = j$. A path with $v_0 = v_k$ is a **cycle**, and a cycle is simple if $v_i = v_j$ for $0 \leq i < j \leq k - 1$. An acyclic graph is called a tree, and a disjoint union of trees is called a forest.

An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be **connected** if every vertex can be reached from every other vertex. A graph consisting of only one vertex is considered

connected. Graphs that are not connected are called disconnected. For a disconnected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a connected component of \mathcal{G} is a induced subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ connected and maximal, that is, there is a subgraph $\mathcal{G}'' = (\mathcal{V}'', \mathcal{E}'')$, so that $\mathcal{V}' \subset \mathcal{V}''$. To check whether a graph is connected and find all of its connected components, it can be used a breadth or depth search, with computational complexity of order $O(V + E)$. Graphs in which the number of edges is linearly comparable to the number vertices are called **sparse graphs**, that is, $O(E) = V$.

2.2.2 Network measures

The complex networks research area have a number of measures to characterize different network structures and properties. In this section, the most relevant measures for this work are described.

Average degree

The average degree of a network is a relatively simple measure that statistically quantifies the average degree of vertices in a network, as is described in section 2.2.1 by Eq. 2.1.

Clustering coefficient

The clustering coefficient is a measure related to network clusters represented by local structures. Basically, this coefficient measures the number of edges forming triangles in a network. Its has high value if two vertices that share a common neighbor vertex have high probability of being connected. Formally, the global clustering coefficient is defined as:

$$c = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples}}, \quad (2.2)$$

where "connected triples" mean a vertex connected to two other different vertices.

A second definition of the clustering coefficient is a local measurement for each vertex v_i . The clustering coefficient for a vertex is defined as the ratio between the number of connected pairs of neighboring vertices and the total amount of possible connections. For an undirected network:

$$c_i = \frac{2|e_{jk}|}{k_i(k_i - 1)}, \quad (2.3)$$

where k_i is the degree of vertex v_i and $e_{jk} = 1$ if the v_i neighbor vertices v_j and v_k are connected, and $e_{jk} = 0$ otherwise. Figure 2.4 depicts a simple example of both

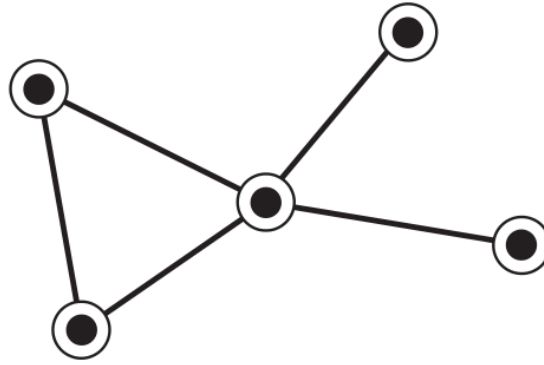


Figure 2.4: A small network to illustrate the clustering coefficient measure. In this network, there are 1 triangle and 8 connected triples resulting in a global clustering coefficient of $3 \times 1/8 = 3/8$. From left to right, the local coefficients are 1, 1, $1/6$, 0 and 0.

measures.

Assortativity

The vertex preferences to connect to other vertices in the network that present similarities or dissimilarities in relation to their degree, in the structural sense, is numerically translated by the assortativity or the assortative mixing measure (Newman, 2003a). It is essentially the Pearson correlation between two linked nodes concerning their degrees. In general, the assortativity r is restricted to the interval $[-1, 1]$. When $r = 1$, it is said that the network has perfect assortativity standards, whereas when $r = -1$, it is said that the network is completely disassortative, which occurs when high-degree vertices tend to connect to low-degree vertices. The network assortativity can be calculated by the following equation:

$$r = \frac{e^{-1} \sum_{e_{ij} \in \mathcal{E}} k_i k_j - \left[e^{-1} \sum_{e_{ij} \in \mathcal{E}} \frac{1}{2} (k_i + k_j) \right]^2}{e^{-1} \sum_{e_{ij} \in \mathcal{E}} \frac{1}{2} (k_i^2 + k_j^2) - \left[e^{-1} \sum_{e \in \mathcal{E}} \frac{1}{2} (k_i + k_j) \right]^2}, \quad (2.4)$$

where k_i and k_j are the degrees of both vertices at each end of edge e_{ij} , and e is the number of edges.

Modularity

Given a network composed of g subgroups of densely connected vertices such that the connections between different subgroups are sparse, the modularity measure (Newman and Girvan, 2004) is defined as:

$$Q = \sum_i^g (g_{ii} - g_i^2) = \text{Tr}(G) - \|G^2\|, \quad (2.5)$$

where G is a symmetric matrix whose element g_{ij} is the fraction of all edges that connect vertices of subnetworks i and j , $\|G\|$ represents the sum of all elements of matrix G , $Tr(G)$ is the trace of the matrix G , and g_i is defined as the column sum $g_i = \sum_j g_{ij}$.

The modularity quantifies the fraction of edges that connect vertices within a same group and subtract the expected value of the same quantity measured on a network hypothetical division with the same groups but considering random connections between the vertices. The trace of the matrix B computes the fraction of edges in the network that connect vertices to the same densely connected subnetwork. Therefore, a good modular division of the network should result in a high value for this calculation. The variable a_i informs about the fraction of edges connecting vertices within the subgroup i . Thus, if the number of connections within a subgroup are no better than random connections, then $Q = 0$. On the other hand, when Q is approaching 1, it can be considered that the network has a clear division concerning the g subgroups.

Betweenness

Betweenness is a centrality measure which quantifies the number of shortest paths passing through a specific vertex or edge in the network. When analyzing the traffic information of a network, for example, it describes the load on a vertex or an edge (Zhao *et al.*, 2005a, 2007). Similarly, it describes the centrality importance of certain vertices or edges in events of cascading failures (Zhao *et al.*, 2004, 2005b). The betweenness measure can be calculated by using the algorithm introduced by Newman (2001):

Algorithm 1 Calculation of vertex betweenness

Input:

\mathcal{G} : a graph

Output:

b_i : betweenness of each vertex v_i

for $i = 1 \rightarrow V$ **do**

1. calculate all the paths from every vertex v_j to v_i by using the width search algorithm;

2. associate to every vertex v_j a variable b_j with initial value equals to 1;

3. starting from the farthest vertex v_j , follow the shortest paths to v_i by adding the current value of b_k to the variable of the next vertex of the path b_{k+1} . When there is more than one vertex at the same stage of the path, the value of b_k is equally divided among the vertices;

end for

4. the betweenness of each vertex is the value of the associate variable b_i

Katz index

Similarly to the betweenness measure, the Katz index (Katz, 1953) is also a centrality measure. However, in this case, the concept of centrality is perceived differently: the more central is a vertex as the more central are its neighbors. Thus, the neighboring vertices directly influence the centrality of a given vertex. As a practical example, it can be considered a voting situation where each person is a network vertex. If only the vertices v_j and v_k vote on the vertex v_i , and all other vertices of the network vote on v_k and v_j then, probably, v_i is the most important vertex.

The influence of a neighboring vertex v_i decreases as the distance from the reference vertex increases. Hence, the Katz measure includes a factor, $\alpha < 1$, for weighting distances. The Katz index of v_i can be written as:

$$c_i^{Katz} = \sum_{k=1}^{\infty} \sum_{j=1}^n \alpha^k (\mathcal{A}^k)_{ji}, \quad (2.6)$$

where \mathcal{A} is the network adjacency matrix and n is the number of vertices. It should be noted that the calculation \mathcal{A}^k_{ij} results in the total weight of paths of length equals to k , considering the weights of edges between vertices v_j and v_i . In matrix form, this equation can be rewritten as:

$$C^{Katz} = \sum_{k=1}^{\infty} \alpha^k (\mathcal{A}^T)^k \mathbf{1}_n, \quad (2.7)$$

where $\mathbf{1}_n$ is a unitary n -dimensional vector. This equation is convergent if α is smaller than the reciprocal of the largest eigenvalue of the adjacency matrix \mathcal{A} , and, in this case, it can be reduced to:

$$C^{Katz} = ((I - \alpha \mathcal{A}^T)^{-1} - I)I, \quad (2.8)$$

where I is the identity matrix of size n .

PageRank centrality

Page rank centrality attempts to model the behavior of visiting web pages (Brin and Page, 1998). Most of the time, Internet users visit Web pages by clicking on hyper-links (network edges) from a page that is currently being visited, or visit the pages through bookmarks and by typing addresses. In a network, this process can be modeled by a combination of a random walk and occasional jumps to certain vertices. This model

can be described by the following relationship:

$$p(i) = \frac{q}{n} + (1 - q) \sum_{j:j \rightarrow i} \frac{p(j)}{k_{outj}}, \quad i = 1, 2, \dots, n, \quad (2.9)$$

where $p(i)$ is the page rank of the vertex v_i , k_{jout} is the degree of vertex v_j considering only the outgoing edges, and the sum is performed on the vertices v_j of which edges connect to v_i . The parameter q weight the mix between random walk and page jumps.

Eigenvalue centrality

This measure is also based on the principle that the significance of a vertex depends on the importance of its neighbors (Phillip and Bonacich, 2007). In this case, the relationship is straightforward: the prestige x_i of vertex v_i is proportional to the sum of the prestiges of its neighbors:

$$\lambda x_i = \sum_{j:j \rightarrow i} x_j = \sum_j \mathcal{A}_{ji} x_j = (\mathcal{A}^T x)_i, \quad (2.10)$$

where it can be noted that x_i is the i -th component of the transposed adjacency matrix A with eigenvalue λ .

2.2.3 Constructing networks from data sets

Many are the ways to construct networks from data sets. In this section, we briefly discuss the most common network construction methods. Given a data set, $\mathcal{X} = \{\mathbf{x}_i, i = 1, \dots, V\}$, each instance \mathbf{x}_i is represented by a network node. The links among nodes are created by one of the following methods.

Fully connected networks: these kind of networks are constructed by inserting links between all pairs of nodes, in a way that similar nodes have large link weights between them. The disadvantage of these networks is the computational cost, given that the network is dense. On the other hand, the advantage of fully connected networks is in weight learning - with a differentiable weight function, one can easily take the derivatives of the graph. Empirically, it has been observed that fully connected networks perform worse than sparse networks (Zhu, 2005).

Sparse networks: differently from fully connected networks, sparse networks have fewer connections by avoiding links between dissimilar nodes. Because the smaller number of links, they are computationally fast. They also present good empirical performance, which can be due to the inexistence of links between dissimilar nodes which usually belong to different classes. The links in such networks can be weighted or unweighted. One disadvantage is weight learning - a change in weight hyperparameters

will likely change the neighborhood, making optimization awkward.

k -NN networks: by following this network construction method, nodes i and j are connected if i is in j 's k -nearest-neighborhood or vice-versa. k is a hyperparameter that controls the density of the network. These networks may result asymmetric because, in the case i is in j 's k -nearest-neighborhood, j may not be in i 's k -nearest-neighborhood. k NN networks present the property of adaptive scales, because the neighborhood region is different in low and high data density regions.

k -Associated Optimal Graphs (kAOG): as an extension of the previous k -NN networks, kAOG is a method developed by Bertini *et al.* (2011) which has been applied to supervised learning. It uses a measure called purity to construct graph components while maximizing this measure. In the process for constructing the kAOG, k is increased 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 is used to compare among components of different k -associated graphs formed with different values of k . The component with the highest purity value is held, and the others are discarded. This method is reviewed with more details in section 2.4.4.

ϵ -NN networks: in this network construction method, there is a link between nodes i and j if the distance (or dissimilarity) between them are smaller than a threshold ϵ , $d(i, j) < \epsilon$. The hyperparameter ϵ controls the neighborhood radius, and, despite being continuous, the search for its optimal value is discrete and the search space size is bounded by the total number of links $O(n^2)$.

Figure 2.5 depicts the resulted network for a 2-dimensional Gaussian data sets by using different construction methods. Figure 2.5a shows the two classes. Figure 2.5b shows the vertices connected to their 3-nearest neighbors of the same class. Figure 2.5c depicts the kAOG, where it can be noted that this construction method has the sparseness property, that is, the resulted network is composed by many isolated components. The ϵ -NN network is showed in Fig. 2.5d, where some isolated components are due to the absence of neighbor vertices in the specific ϵ -region.

2.3 Dynamical processes on networks

This section gives a brief review of the main dynamical processes on networks related to this thesis.

2.3.1 Random walk

The basics of the random walk theory are revised in this subsection. Random walks can be understood in terms of Markov chains (Gallager, 1995). It is usual to think of a Markov chain as the sequence of states entered by a system evolving in time, or the se-

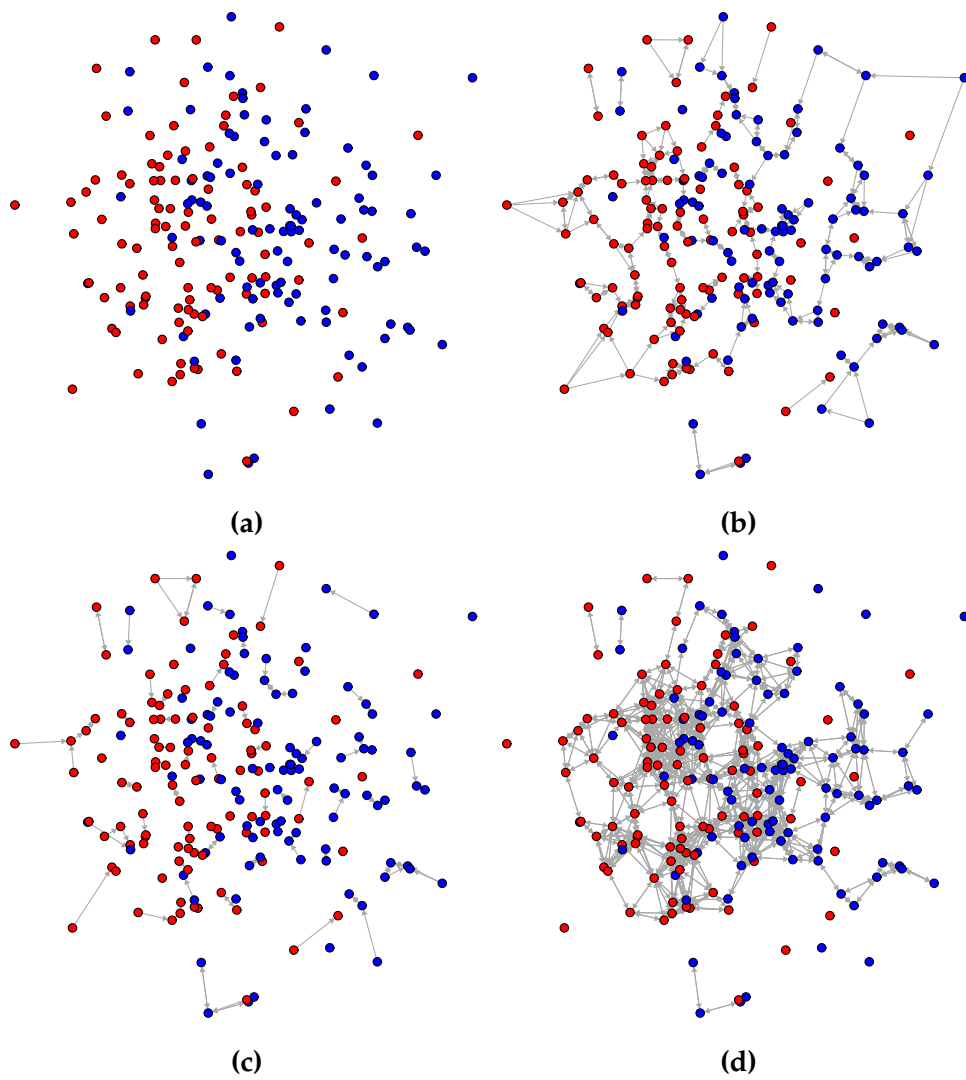


Figure 2.5: Examples of networks constructed by using different methods. (a) Artificial data set composed of two mixed Gaussian-distributed labeled sets. (b) Network constructed by using the k -NN technique with $k = 3$. (c) Network constructed by the kAOG method. (d) Network constructed by using the ϵ -radius technique, where $\epsilon = 30\%$ of the average distance among all vertices.

quence of positions occupied by a moving particle. The theory of Markovian processes comprises the largest and the most important chapter in the theory of stochastic processes; this importance is further enhanced by the many applications it has found in the physical, biological, and social sciences, and in engineering and commerce (Çınlar, 1975).

Consider a stochastic process Ω with a finite state space Γ , and P a probability measure on it. For each $n \in \mathbb{N} = \{0, 1, 2, \dots\}$, $\Omega_n \in \Omega$ is an element from Γ . It is customary to say the process is in state j at time n to mean $\Omega_n = j$. The stochastic process $\Omega = \{\Omega_n\}$ is called a Markov chain if $P(\Omega_{n+1} = i | \Omega_0, \dots, \Omega_n) = P(\Omega_{n+1} = i | \Omega_n)$, $i \in \Gamma$, that is, the process is independent of past states provided that the current state Ω_n is known. In this work, time-homogeneous chains are considered, that is, when $P(\Omega_{n+1} = j | \Omega_n = i) = p_{ij}$ is independent of n . The probabilities p_{ij} can be arranged into a Markov square matrix $\mathcal{P} = \{p_{ij}\}$.

\mathcal{P} is called a Markov matrix provided that: i) for any $i, j \in \Gamma$, $P(i, j) \geq 0$; and ii) for each $i \in \Gamma$, $\sum_{j \in \Gamma} P(i, j) = 1$. Thus, the transition matrix of a Markov chain is a Markov matrix. The probability that the chain moves from state i to state j in m steps is the (i, j) -entry of the m th power of the transition matrix \mathcal{P} . For any $m, n \in \mathbb{N}$:

$$\mathcal{P}^{m+n}(i, j) = \sum_{k \in \Gamma} \mathcal{P}^m(i, k) \mathcal{P}^n(k, j),$$

which is called the Chapman-Kolmogorov equation. It states that starting at state i , in order for the process Ω to be in state j after $m + n$ steps, it must be in some intermediate state k after the m th step and then move from that state k into state j during the remaining n steps.

The states of a Markov chain can be classified concerning some properties. Let T be the time of first visit to state j , and let N_j be the total number of visits to state j . Some of the properties are as follows:

- state j is called *recurrent* if $P_j\{T < \infty\} = 1$; otherwise, if $P_j\{T < +\infty\} > 0$, then j is called *transient*;
- a recurrent state j is called *null* if $E_j[T] = \infty$; otherwise, it is called *non-null*;
- a recurrent state j is said to be *periodic* with period δ if $\delta \geq 2$ is the largest integer for which $P_j\{T = n\delta \text{ for some } n \geq 1\} = 1$; otherwise, if there is no such $\delta \geq 2$, j is called *aperiodic*.
- a set of states is said to be *closed* if no state outside it can be reached from any state in it;
- a state forming a closed set by itself is called an *absorbing* state;

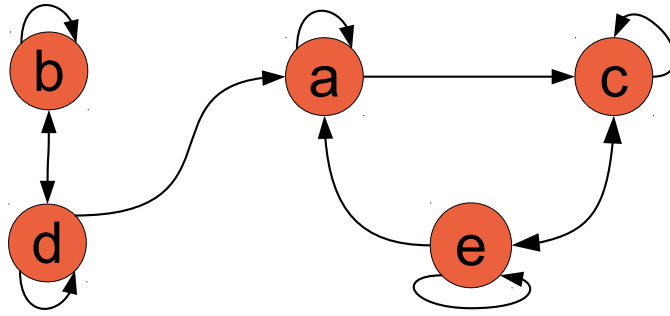


Figure 2.6: Transition graph of the Markov matrix in Eq. 2.11. Vertices represent states, and a directed link from state i to state j exists if $p_{ij} > 0$.

- a closed set is *irreducible* if no proper of it is closed;
- a Markov chain is called *irreducible* if its only closed set is the set of all states.

In other words, a Markov chain is irreducible if and only if all states can be reached from each other. As an example, consider the following Markov matrix:

$$\mathcal{P} = \begin{array}{cc} & \text{states} \\ \begin{bmatrix} 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 2/5 & 0 & 3/5 \\ 1/2 & 1/4 & 0 & 1/4 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \end{bmatrix} & \begin{array}{l} a \\ b \\ c \\ d \\ e, \end{array} \end{array} \quad (2.11)$$

which is composed of five states, namely a , b , c , d and e . For an easy visualization, the above matrix can be mapped into the network depicted in Fig. 2.6, where vertices represent states, and a link between states i and j exists if $p_{ij} > 0$. From this figure we see that from set $\{b, d\}$ the set $\{a, c, e\}$ can be reached but not vice-versa. Thus, once the process leaves the states $\{b, d\}$, neither b nor d can ever be visited again. The closed sets are $\{a, b, c, d, e\}$ and $\{a, c, e\}$ and, since there are more than one closed set, this chain is not irreducible. Deleting the second and the fourth rows and columns (set $\{b, d\}$), we have:

$$\mathcal{P}' = \begin{array}{cc} & \text{states} \\ \begin{bmatrix} 1/4 & 3/4 & 0 \\ 0 & 2/5 & 3/5 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} & \begin{array}{l} a \\ c \\ e, \end{array} \end{array}$$

which is the Markov matrix \mathcal{P} restricted to the closed set $\{a, c, e\}$.

If all states are rearranged, the two closed sets can be easily identified:

$$\mathcal{P} = \begin{array}{cc} & \text{states} \\ \left[\begin{array}{ccccc} 1/4 & 3/4 & 0 & 0 & 0 \\ 0 & 2/5 & 3/5 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 1/4 & 1/4 \end{array} \right] & \begin{array}{l} a \\ c \\ e \\ b \\ d, \end{array} \end{array}$$

where states a , c and e are recurrent non-null aperiodic, and states b and d are transient.

The probability for a random walker starting at state i_0 end at a state i_m is given by the probability of the chain $P(i_0, i_m) = P(i_0, i_1)P(i_1, i_2) \dots P(i_{m-1}, i_m)$. If an infinite number of transitions, $m \rightarrow \infty$, is considered, then the **limiting probabilities** need to be calculated, also called **invariant distributions** or **steady states**. It can be shown that the limiting probability $\mathcal{P}^\infty(j) = \lim_{n \rightarrow +\infty} \mathcal{P}^n(i, j)$ exists given a recurrent, non-null and aperiodic state j , and the limiting probability of the final state j is independently of the initial state i (Çinlar, 1975). Moreover, if Ω is an irreducible aperiodic Markov chain with finitely many states, then

$$\mathbf{p}\mathcal{P} = \mathbf{p}, \quad (2.12)$$

has a unique solution, where \mathbf{p} is a normalized row vector, $\mathbf{p}\mathbf{1} = 1$, and $\mathbf{p}(j)$ is the limiting probability of state j , $\mathbf{p}(j) = \mathcal{P}^\infty(j)$.

The limiting probabilities can be calculated by finding the eigenvector corresponding to the unit eigenvalue of matrix $\mathcal{P}^\mathcal{T}$, the transpose of matrix \mathcal{P} , with computational cost of $O(n^3)$ due to matrix inversion, or by iterating the system

$$\mathbf{p}_{i+1} = \mathcal{P}^\mathcal{T} \mathbf{p}_i \quad (2.13)$$

to the stationary state, with computation cost of $O(n^2)$.

2.3.2 Tourist walk

Similarly to random walk (section 2.3.1), a tourist walk is composed of a walker (tourist) aiming at visiting sites (network vertices) in an underlying network. 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, where the self-avoiding factor is limited to the memory window size $\mu - 1$. Therefore, it is prohibited that a trajectory to intersect itself inside this memory window. In spite of being a simple rule, it has been

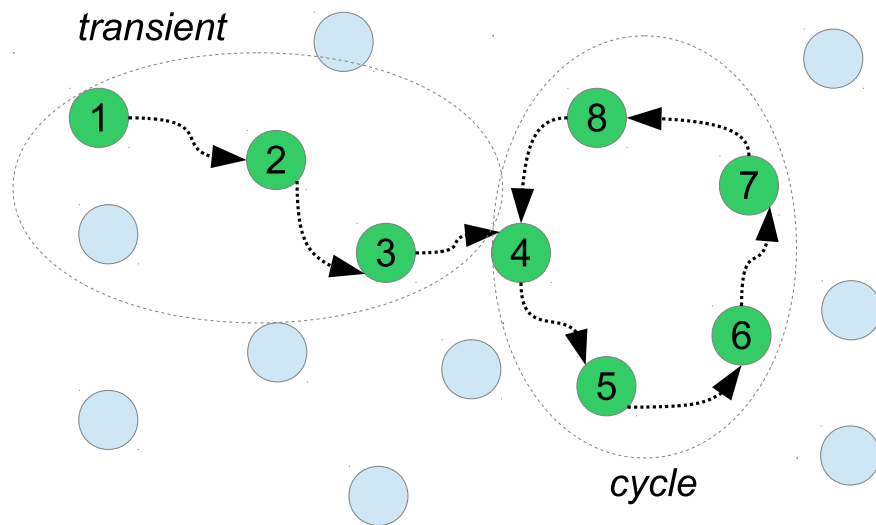


Figure 2.7: A simple example of a tourist walk with memory window size $\mu = 1$, transient length $t = 3$ and cycle length $c = 5$.

shown that this kind of movement possesses complex behavior when $\mu > 1$ (Lima *et al.*, 2001).

A tourist walk can be decomposed into two distinct terms: the initial *transient* part of length t , and the *cycle* part (*attractor*) with period c . For illustration, Fig. 2.7 depicts a simple example of a tourist walk with $\mu = 1$, transient part length $t = 3$, and cycle part period $c = 5$. However, the attractor or cycle period is not easily detectable. Given that a cycle begins and ends at a same vertex, one may think that, if the walker revisit a vertex, a cycle is configured. But this simple reasoning is probably a wrong approach for detecting attractors. In fact, during a walk, a vertex may be revisited without configuring an attractor, and the tourist's finite memory μ allows revisitations to non-attractor vertices. For instance, if we choose $\mu = 5$ for the network in Fig. 2.7, the revisitation performed by the tourist on the vertex 4 would have not configured an attractor, since the site 5 would still be in the memory μ and so forbidden to be revisited, and the tourist would be compelled to visit another vertex. This characteristic enables sophisticated trajectories through the network, at cost of increasing the difficulty to detect attractors.

The possibility of a tourist to visit any other site other than the ones contained in its memory window is considered in most of the works related to tourist walks (Kinouchi *et al.*, 2002; Lima *et al.*, 2001; Stanley and Buldyrev, 2001). As memory window μ increases, the chance of the walker to perform large jumps in the data set also increases, since its neighborhood is most likely to be already entirely visited within the time frame μ . In situations in which this characteristic is undesired, a specific network structure can be constructed *a priori*, and the walker is only permitted to visit vertices

that are in its directly connected neighborhood. With this mechanism, it also can occur that, for large values of μ , the walker gets trapped within a vertex, being unable to visit other vertices of the neighborhood. In this scenario, we say that the walk has a transient part t and a null cycle period ($c = 0$).

2.3.3 Chaotic synchronization

Chaotic synchronization is a basic feature in nonlinear science (Pikovsky and Kurths, 2001). It is defined as the complete coincidence of the trajectories of coupled individual chaotic systems in the phase space, where each individual system is represented by a network vertex. Mathematically, given state variable vectors \mathbf{x} and \mathbf{y} representing two dynamical systems, they are said to be completely synchronized if $|\mathbf{x}(t) - \mathbf{y}(t)| \rightarrow 0$ as $t \rightarrow \infty$. Since the early 1990s, there have been strongly increasing interests in the synchronization of chaotic systems. The phenomenon was first discovered by Fujisaka and Yamada (Fujisaka and Yamada, 1983). The seminal paper by Pecora and Carroll (Pecora and Carroll, 1990) triggered much interest in this topic (Pikovsky and Kurths, 2001). Up to now, chaotic synchronization has been extensively studied by researchers of applied sciences, such as electrical and mechanical engineering, biology and laser systems, etc. Mathematical methods to study synchronization among coupled chaotic systems were presented in (Gameiro and Rodrigues, 2001) and (Rodrigues *et al.*, 2001), and studies on synchronization of a large number of coupled chaotic elements in general network topologies were performed in (Belykh *et al.*, 2006).

In Zhao *et al.* (2008a), the authors considered the synchronization role in networks of general topology of coupled continuous chaotic elements with parameter mismatch. Each element in the network can be one of a class of chaotic oscillators, which has a stable linear part perturbed by a bounded nonlinear function. Sufficient conditions to obtain synchronization of the oscillators are as follows. Consider the following system:

$$\dot{\mathbf{x}}_i = A\mathbf{x}_i + f(t, \mathbf{x}_i, \mu_i) + \kappa \sum_{j=1}^n \gamma_{ij}(\mathbf{x}_j - \mathbf{x}_i), \quad i = 1, \dots, n, \quad (2.14)$$

$$\|f(t, \mathbf{x}_i, \mu_i)\| \leq L_1,$$

$$\max\{f'(t, \mathbf{x}_i, \mu_i)\} = L_2,$$

where μ_i is a parameter of vertex i , the real part of the largest eigenvalue of A is negative, $Re(\lambda_{max}) < 0$, λ_{max} is the largest eigenvalue of matrix A , $\kappa \geq 0$ is the coupling strength, L_1 and L_2 are positive constants, n is the number of oscillators in the network

and γ_{ij} defines whether there is a connection between oscillator i and j :

$$\gamma_{ij} = \begin{cases} 1, & \text{if element } i \text{ is connected to } j, \\ 0, & \text{otherwise.} \end{cases}$$

The synchronization analysis with parameter mismatch requires that the solution of the coupled system is bounded. In such a network, the Neumann boundary conditions are assumed. All elements represented by Eq. 2.14 with arbitrary γ_{ij} configuration can be synchronized by providing sufficiently large coupling strength. The basic idea to prove boundedness consists in constructing a Lyapunov function $U(\mathbf{x})$ and showing that $\dot{U}(\mathbf{x}) \leq 0$ as $U(\mathbf{x}) = D > D_0$ where D_0 is a sufficiently large positive constant (Afraimovich *et al.*, 1997). Then, the synchronization state ($\mathbf{x}_1 = \mathbf{x}_2 = \dots = \mathbf{x}_n$) is asymptotically stable if the following condition holds:

$$\kappa > \frac{1}{n} \left(\|A\| + L_1 + \frac{\rho}{2} \right) \sum_{j>i, q \in P_{ij}}^n \Omega(P_{ij}). \quad (2.15)$$

where ρ is a positive constant, P_{ij} is the set of vertices on a shortest path connecting nodes i and j , and $\Omega(P_{ij})$ is the length of the shortest paths.

2.3.4 Consensus and pinning control

In a network of coupled dynamical systems, the term "consensus" stands for reaching an agreement regarding a certain quantity of interest that depends on the states of all agents (vertices) (Olfati-Saber *et al.*, 2007). The basic idea of consensus is that each agent updates its own state based on the states of its neighbors. At the end of the process, all agents reach a common value (Chen *et al.*, 2009a).

Let $\mathcal{A} = [a_{ij}]$ be the adjacency matrix of a network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, V\}$ is the set of vertices and $\mathcal{E} = \{1, \dots, E\}$ is the set of edges. If there is an edge between vertices i and j , then $a_{ij} \neq 0$. If there is no edge, $a_{ij} = 0$. The set of direct neighbors of vertex i is described by $\lambda_i = \{j \mid a_{ij} \neq 0\}$. The continuous evolution rule for the state of each vertex i is defined by the following equation:

$$\dot{x}_i(t) = \sum_{j \in \lambda_i} a_{ij} [x_j(t) - x_i(t)], \quad (2.16)$$

where $x_i(t)$ represents the dynamics of vertex i .

The linear system represented by Eq. (2.16) is the distributed consensus algorithm proposed in (Olfati-Saber and Murray, 2004). This system converges to a common state via local interactions. Assuming that the network is undirected, the sum of the states of all vertices is invariant and the consensus is reached asymptotically; thus, the

collective decision α is equal to the average of the initial states of all vertices, that is, $\alpha = \frac{1}{V} \sum_{i=1}^V x_i(0)$. Consensus algorithms with such a property of invariant sum of the states of the vertices are called **average-consensus** algorithms.

There have been attempts to control the dynamics of a complex network to an arbitrarily desired state, as an equilibrium point or a periodic orbit of the network (Xiang *et al.*, 2007). One of the approaches, called **pinning control**, consists in injecting only a small number of local feedback controllers in the network. Considering that the network \mathcal{N} represents a communication network of n coupled oscillators, the continuous time evolution of the i th oscillator can be described by:

$$\dot{x}_i(t) = f(x_i(t)) - c \sum_{j=1}^V a_{ij} h[x_j(t) - x_i(t)], \quad (2.17)$$

where $x_i(t)$ represents the state vector of the i th oscillator, $f(x_i(t))$ describes the oscillator's individual dynamics, $c > 0$ is the overall coupling strength and h is the inner coupling function that characterizes the interactions between neighboring oscillators.

In pinning control, only a small fraction of vertices in the network is expected to be controlled. Without loss of generality, let the first p vertices be selected to be pinned. Thus, adding control and using the Laplacian matrix¹ L , Eq.(2.17) can be rewritten as follows:

$$\dot{x}_i(t) = f(x_i(t)) - c \sum_{j=1}^n l_{ij} h(x_j(t)) + u_i(t), \quad (2.18)$$

where $u_i(t) = g_i \{h[s(t) - x_i(t)]\}$ is the local feedback controller, $s(t)$ is the reference trajectory and g_i quantifies the pinning control gain for vertex i . Derivation from Eq. (2.17) to Eq. (5.1) requires that function h be linear, as it is considered in this work. The reference trajectory is described by an independent oscillator $\dot{s}(t) = f(s(t))$. Therefore, $g_i = 0$ if vertex i is not pinned. The task here is to drive the dynamic complex network to $s(t)$ as $t \rightarrow \infty$ by pinning some vertices.

2.4 Relevant network-based techniques in machine learning

In this section, we describe some of the most relevant techniques related to the works presented in this thesis. All techniques share one important aspect: they are network-based. The application areas range from unsupervised to supervised learning, including semi-supervised extensions. The main concepts involved are random walk, label propagation, dynamical synchronization and network construction. These

¹Network \mathcal{N} can be described via its Laplacian matrix $L = [l_{ij}] = \mathcal{D} - \mathcal{A}$, where $\mathcal{D} = \text{diag}(d_1, \dots, d_n)$ is the degree matrix of \mathcal{N} and $d_i = \sum_{j \in \eta_i} a_{ij}$.

concepts served as inspiring ideas for the techniques presented in the next chapters.

We start with a brief of the vast applications of random walk theory. In image analysis, for example, a random walk process can be executed through pixels represented by network nodes. Texture discrimination and edge detection were performed by comparing boundary distributions of such a process (Wechsler and Citron, 1980; Wechsler and Kidode, 1979). As an alternative way to perform edge detection and image segmentation, other measures were derived. In (Grady, 2006), first time passage probabilities are computed when a random walker passes through a labeled node (pixel) after starting from an unlabeled node. The probabilities are compared and the label with the highest probability is assigned to the unlabeled pixel. A similar approach, applied to content-based image retrieval, can be found in (Bulò *et al.*, 2011), in which relevant and non-relevant images labeled by a specialist are seed nodes. A ranking score for each unlabeled image is computed from the probability that a random walker starting at that image will reach a relevant seed before encountering a non-relevant one. Random walk has also been extensively applied to unsupervised learning, such as community detection (Fortunato, 2010) and data clustering. In the next subsection, a specific clustering technique based on random walk and particle competition is detailed.

2.4.1 Particle competition for unsupervised and semi-supervised learning

In (Quiles *et al.*, 2008; Silva and Zhao, 2012a,b; Silva *et al.*, 2013), random walk is used in combination with preferential walk in competitive processes. Competing particles walk through nodes in a network and the number of passages of a particle through a node determines its domination over that node. The cluster to which a node pertains is determined by the particles with the highest domination.

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:

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

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

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:

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

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

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. A rigorous model of particle competition via a stochastic competitive dynamical system is introduced in Silva and Zhao (2012b); Silva *et al.* (2013).

As an illustrative example, consider Fig. 2.8, where there are $M = 4$ communities. The black stars represent vertices that do not have an owner particle. $K = 4$ particles are inserted in a random manner (following a uniform distribution), namely the red (circle-shaped), the blue (square-shaped), the green (diamond-shaped), and the magenta (triangle-shaped). Figure 2.8a shows the initial particle placement. The ownership of a vertex is given by its color and shape. Figure 2.8b shows the ownership of the vertices after 300 iterations, Fig. 2.8c, after 800 iterations, and Fig. 2.8d, after 1700 iterations, when all 4 communities have been properly discovered.

In Silva and Zhao (2012a), the authors extend the unsupervised particle competition to a semi-supervised version. 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 semi-supervised learning version does have some external knowledge by definition. The main difference of the semi-supervised version is that each particle 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. In this case, each particle always represents a labeled vertex, called the home vertex. In the reanimation procedure of a particle, it no more 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.

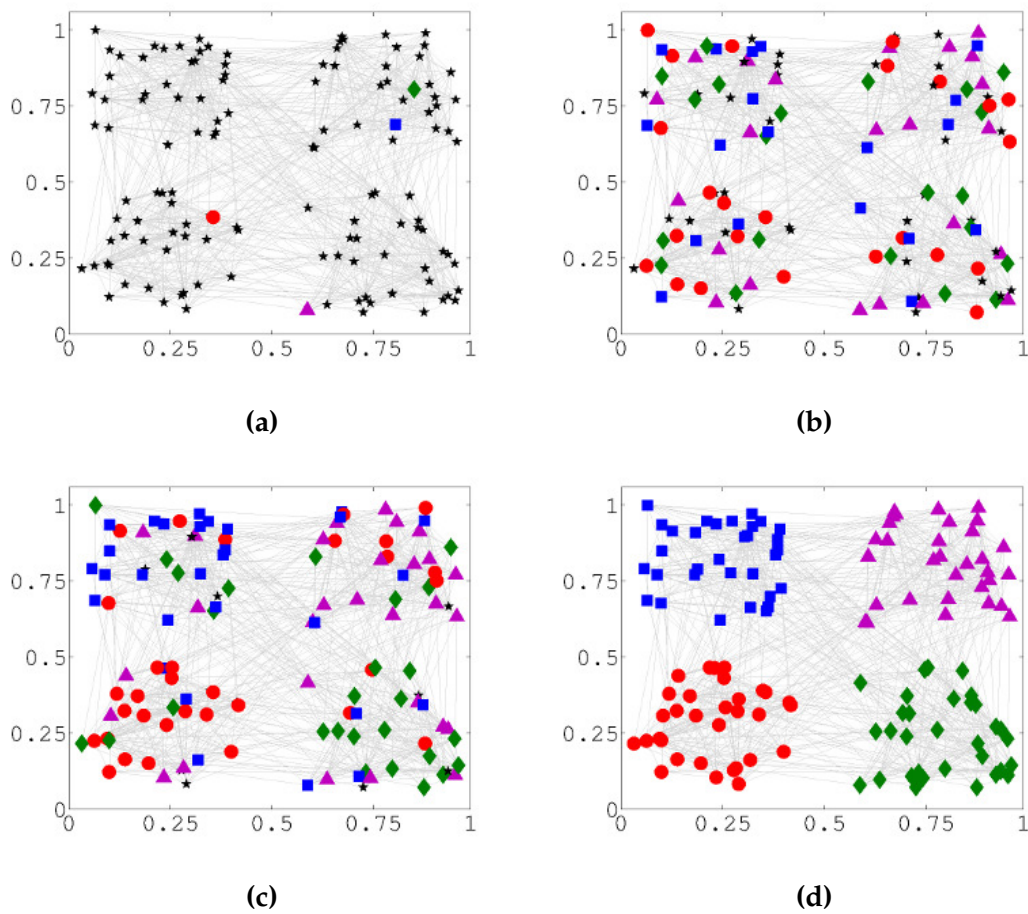


Figure 2.8: Illustration of community detection via particle competition in an artificial network. The total number of vertices is $n = 128$, and the total number of balanced communities is $M = 4$, $\langle k \rangle = 16$, and $z_{\text{out}}/\langle k \rangle = 0.3$. $K = 4$ particles are inserted in a random manner. Snapshot of the network when: (a) $t = 0$; (b) $t = 300$; (c) $t = 800$; and (d) $t = 1700$. Figure extracted from Silva and Zhao (2012b).

2.4.2 Unsupervised scene segmentation via synchronization of locally coupled chaotic oscillators

The chaotic synchronization studies introduced in subsection 2.3.4 have been applied by Zhao *et al.* (2008a) to the scene segmentation problem. In scene segmentation, each object in a given scene can be of any form. Consequently, the corresponding oscillators representing each object may have arbitrary connection topology, which means that the synchronization conditions obtained on regular lattice (1D, 2D, or even higher dimension) do not cover the segmentation application. Zhao *et al.* (2008a) consider the synchronization role in networks of general topology of coupled continuous chaotic elements with parameter mismatch. Neumann boundary condition is assumed. Each element in the network can be one of a class of chaotic oscillators, which has a stable linear part perturbed by a bounded nonlinear function. A network of coupled Wilson-Cowan neural oscillators is used to solve image segmentation problems by rapid chaotic synchronization and desynchronization.

The segmentation strategy introduced by Zhao *et al.* (2008a) is as follows. Consider a scene image containing several objects. A network is constructed so that each element corresponds to a pixel of the image. As the system runs, the vertices or oscillators self-organize according to a predefined similarity criterion as, for example, the connections between pairs of neighboring oscillators with similar gray level or color are kept, while oscillators of very different gray level or color are not connected to each other. Consequently, all vertices belonging to the same segment will be synchronized to form a unique trajectory, and each object is represented by a synchronized chaotic orbit. In this way, objects can be segmented.

The model for scene segmentation is a two dimensional network governed by the following equations:

$$\begin{aligned} \dot{x}_{i,j} &= -ax_{i,j} + G(cx_{i,j} + ey_{i,j} + I_{i,j} - \theta_x) + k \sum_{(p,q) \in \Delta_{i,j}} \gamma_{p,q;i,j} (x_{p,q} - x_{i,j}) \quad (2.19) \\ \dot{y}_{i,j} &= -by_{i,j} + G(dx_{i,j} + fy_{i,j} - \theta_y) + k \sum_{(p,q) \in \Delta_{i,j}} \gamma_{p,q;i,j} (y_{p,q} - y_{i,j}) \\ G(v) &= \frac{1}{1 + e^{-(v/T)}} \end{aligned}$$

where (i, j) and (p, q) are lattice points, k is the coupling strength, and $\Delta_{i,j} = \{(i-1, j-1), (i-1, j), (i-1, j+1), (i, j-1), (i, j+1), (i+1, j-1), (i+1, j), (i+1, j+1)\}$. $\gamma_{i,j;p,q} = 1$ if element (i, j) is coupled to (p, q) . Otherwise, $\gamma_{i,j;p,q} = 0$.

Figure 2.9 shows a color image containing eight objects. These patterns are simultaneously presented on a network where each pixel is represented by an oscillator. The initial conditions of all of the oscillators on the grid are random. The coupling strength



Figure 2.9: Sample image for segmentation. Figure extracted from Zhao *et al.* (2008a).

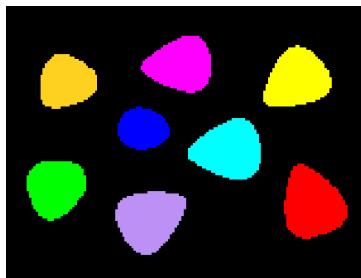


Figure 2.10: Segmentation result when Fig. 2.9 is an input. Figure extracted from Zhao *et al.* (2008a).

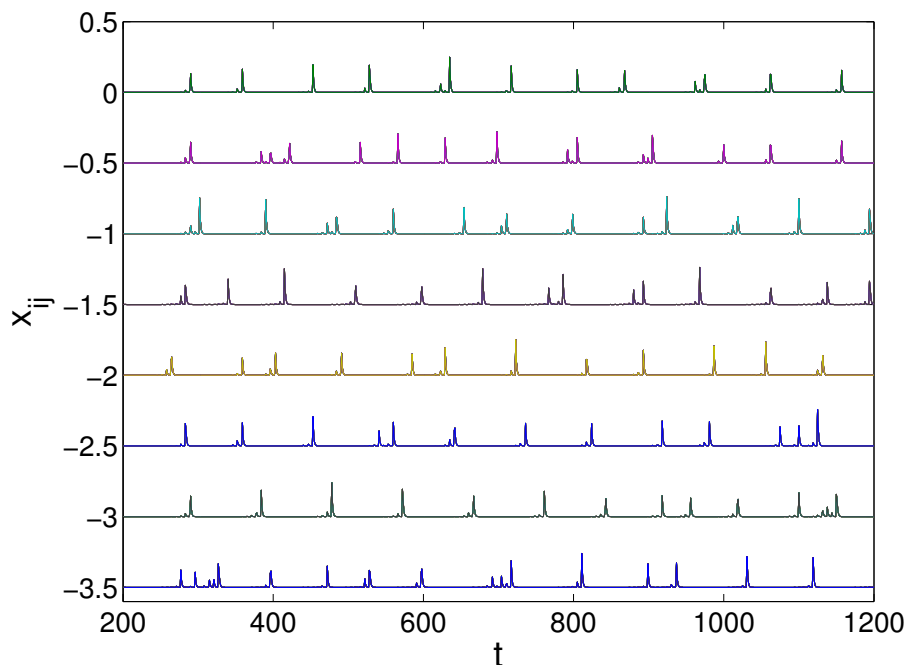


Figure 2.11: Temporal activities of oscillator blocks when Fig. 2.9 is an input. Each trace in the figure is a synchronized chaotic orbit corresponding to an object in the input pattern. Vertical scale of second to eighth oscillator blocks (a group of synchronized oscillators) are shifted downwards by 0.5. The coupling strength is set to $k = 2$. Figure extracted from Zhao *et al.* (2008a).

is $k = 2$. Here, we employ the following simple segmentation criteria, although more complex rules can be projected to get better segmentation results by modifying the $\gamma_{i,j;p,q}$ terms. In this simulation, two oscillators, say (i, j) and (p, q) , which represent two pixels in the image, will be coupled together ($\gamma_{i,j;p,q} = 1$) if and only if the gray-level or color difference between them is less than a constant (30 is used in this simulation). Otherwise, the coupling between them is cut ($\gamma_{i,j;p,q} = 0$). Figure 2.10 shows the segmentation result, while Fig. 2.11 shows the temporal activities of the oscillator blocks. We can see the appearance of eight synchronized chaotic orbits, each one representing an object.

2.4.3 Semi-supervised label propagation: local and global consistency

Label propagation methods typically assume that the data lie on a low-dimensional manifold from a high-dimensional space. They rely upon the smoothness assumption, which states that if two data samples are close, then their labels should be close as well. The main idea of these methods is to build a graph that captures the geometry of this manifold as well as the proximity of the data samples. Test samples are labeled by the propagation of the labels of the labeled data along this manifold, while making use of the smoothness property. The pioneer network-based technique proposed by Zhou *et al.* (2003) has the problem showed in Fig. 2.12 as one of its main motivations. Traditional techniques such as k -NN and SVM (Fig. 2.12(b) and (c), respectively) search for spherical-shaped classes and fail to learn different shapes, which could be correctly detected by network-based techniques.

In their work, Zhou *et al.* (2003) consider a set of $(n \times L)$ -dimensional matrices \mathcal{M} composed of non-negative entries. A matrix $F = [F_1^T, \dots, F_n^T]^T \in \mathcal{M}$ associates each unlabeled item $\mathbf{x}_i^{(u)}$ to a corresponding 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 associates each unlabeled instance $\mathbf{x}_i^{(u)}$ to the maximum value of $F_j, j \in \mathcal{L}$. Moreover, it is used a $(n \times L)$ -dimensional matrix Y where $Y_{ij} = 1$ if the labeled instance $\mathbf{x}_i^{(l)}$ is associated to label $y_i = j \in \mathcal{L}$, and $Y_{ij} = 0$, otherwise. The computational steps of the technique is as follows:

1. generate the affinity matrix W given by $W_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$ 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 and $D(i, i) = \sum_j^n W(i, j)$;
3. iterate the system $F(t+1) = \alpha SF(t) + (1 - \alpha)Y$ until convergence, where $\alpha \in$

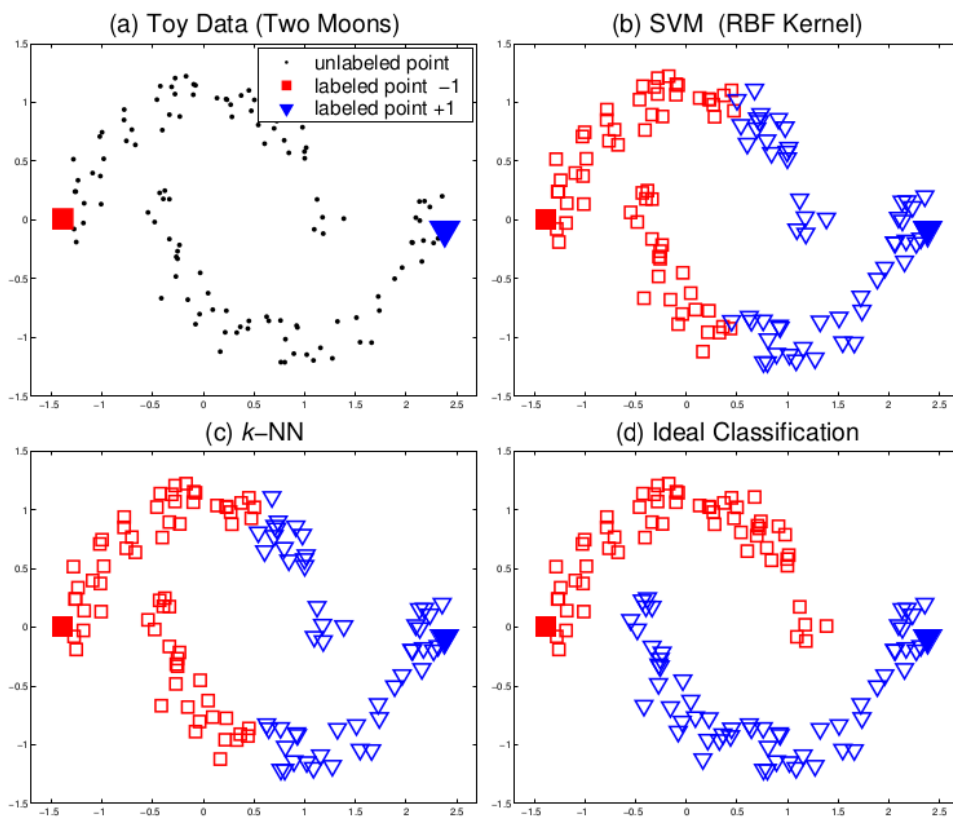


Figure 2.12: The two-moons problem setting to illustrate semi-supervised classification. (a) Two unlabeled classes with just one labeled sample each. (b) Incorrect result achieved by SVM with RBF kernel. (c) Incorrect result achieved by k -NN technique. (d) Ideal classification result. Figure extracted from Zhou *et al.* (2003).

(0, 1);

4. being F^* the limit of the sequence $\{F(t)\}$, each unlabeled vertex $x_i^{(u)}$ is associated to the label $y_i = \arg \max_{j \in \mathcal{L}} F_{ij}^*$.

Moreover, it has been shown in Zhou *et al.* (2003) that the sequence $\mathcal{S} = \{F(t)\}$ is convergent and can be promptly calculated by the following equation:

$$F^* = \lim_{t \rightarrow \infty} F(t) = (I - \alpha S)^{-1} Y. \quad (2.20)$$

Furthermore, the authors have found a regularization framework that satisfies the aforementioned dynamics. This framework aims at minimizing a cost function or an energy expression. It is given by:

$$C(F) = \frac{1}{2} \left(\sum_{i=1, j \in \mathcal{L}}^n W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^n \|F_i - Y_i\|^2 \right), \quad (2.21)$$

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

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

Equation (2.21) can be analyzed in function of its two distinct terms. The first term enforces a smoothness decision of the classifier, meaning that a good classification function must not have large derivatives in high-density areas. This defines a **regularizer function**, which 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. On the other hand, the second term symbolizes the adjustment restriction, revealing that a good classification function should not exchange the labels from already labeled data. This constraint determines the **loss function**, which leads the algorithm to penalize decisions that flip the labels of previously labeled vertices. Practically, to minimize this term, it is enough to prevent the change of those vertices. Finally, the parameter μ is responsible for balancing the weight between these two opposite conditions.

In this technique, the propagation is performed by a linear update rule, and the convergence analysis have been fully described. However, the linear propagation rule may mislead correct treatment of nonlinearities present in data. Moreover, since a matrix inversion is required to find the optimal solution, the algorithm computational complexity is bounded by $O(n^3)$, resulting unfeasible for large-scale networks.

Kokiopoulou and Frossard (2010) proposed an application of the label propagation algorithm to the discrete problem of multiple observation sets. The problem is to assign multiple observations of the test objects to a single class of objects. It can be viewed

as a special case of semi-supervised learning, where the unlabeled data represents the multiple observations with the extra constraint that all unlabeled data examples belong to the same class (Kokopoulou and Frossard, 2010). The goal is then to estimate the single unknown class, while generic semi-supervised learning problems attribute the test examples to different classes.

The discrete algorithm called Manifold-based Smoothing under Constraints (MASC) can be summarized by the Alg. 2, where the weight matrix H is calculated by some sort of distance function among data samples and matrix F associates each unlabeled item to a corresponding label. In matrix Y , $Y_{ij} = 1$ if the labeled instance $\mathbf{x}_i^{(l)}$ is associated to label $y_i = j \in \mathcal{L}$, and $Y_{ij} = 0$, otherwise. Vector e_l is the l th canonical basis vector and $\mathbf{1} \in \mathbb{R}^m$ is the vector of ones. The step 5 can be viewed as a discretization of Eq. 2.21. Actually, the search space in the **for loop** is small because it consists of the following \mathcal{L} vectors ($\in \mathbb{R}^m$): $[1, 0, 0, \dots, 0]$, $[0, 1, 0, \dots, 0]$, $[0, 0, \dots, 1, 0]$, \dots , $[0, 0, \dots, 0, 1]$. Since all test samples belong to the same class, the optimal solution can be obtained with a full search, as long as the number of classes stays reasonable (Kokopoulou and Frossard, 2010). The computational cost of this method is $O(n^2 + km\mathcal{L})$.

Algorithm 2 Manifold-based smoothing under constraints

Input: $\mathcal{X}^{(l)}$: labeled data $\mathcal{X}^{(u)}$: unlabeled observations set $|\mathcal{L}|$: number of classes m : number of unlabeled observations n : number of labeled data samples**Parameters:** k : number of nearest neighbors for network construction**Output:** \hat{l} : estimated class for the unlabeled observations**Initialization:**1. construct a k -NN network $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ 2. compute the weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$ and the diagonal matrix D , where $D_{ii} = \sum_{j=1}^n W_{ij}$ 3. compute $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ **for** $l = 1 \rightarrow |\mathcal{L}|$ **do**

4. $F = \begin{bmatrix} Y_l \\ \mathbf{1} e_l^T \end{bmatrix}$

5. $q(l) = \sum_{i \leq \mathcal{L}; j > \mathcal{L}} S_{ij} \|F_i - F_j\|^2 + \sum_{i < \mathcal{L}; j \leq \mathcal{L}} S_{ij} \|F_i - F_j\|^2$

end for6. $\hat{l} := \operatorname{argmin}_p q(l)$

2.4.4 Supervised k-associated graphs

The nonparametric technique for network-based supervised learning known as k-associated graphs has been proposed by Bertini *et al.* (2011). The proposed technique uses the k-Associated Optimal Graph (kAOG), whose construction relies on two concepts: (i) a **purity measure** which, in short, uses the graph representation to measure mixing levels of the original data set samples regarding its classes given a k-neighborhood; and (ii) the **k-associated graph**, which can be considered as an improved adaptive k-nearest neighbor graph. Given a set of labeled vector-based data as the training set, the training process consists of building the kAOG representing the data set as a sparse graph in which components (a connected subgraph) carry local information about the underlying data distribution. This graph structure is used by the kAOG classifier to estimate the probability an unlabeled sample belongs to a given component, and thus inferring its class.

The process for building the kAOG is divided into three steps. First, the technique represents the training set as a directed graph (network), 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. To create this graph, each vertex is connected to its neighbors that lie in a k-neighborhood and belong to the same vertex's class. This particular way to wire vertices produces a directed graph which will be used to define a purity measure. At the end of this process, for a given *k*, each class is represented by one or more components, in other words, the number of components is larger than or equal to the number of data classes presented in the training set. The second step is responsible for computing the purity measure for each graph component. Given the parameter *k*, a vertex can receive $2k$ connections at most. 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 *et al.*, 2011), the networks are considered as digraphs. Therefore, it is expected that vertices have degrees ranging from *k* to $2k$. The purity measure quantifies the proportion of edges that has effectively been created among vertices from the same class over the total number of possible edges, $2k$. In mathematical terms, the purity ϕ of component α , ϕ_α , is defined as:

$$\phi_\alpha = \frac{D_\alpha}{2k}, \quad (2.22)$$

where D_α denotes the average degree of component α . A purity value close to 1 indicates that a large portion of edges exist among vertices in a class component, resulting in high compactness of that component. Lower values reveal high mixing between data from different classes in a k-neighborhood and, in the extreme case where $\phi_\alpha = 0$, a vertex in that component does not make any connections and is kept isolated. Figure

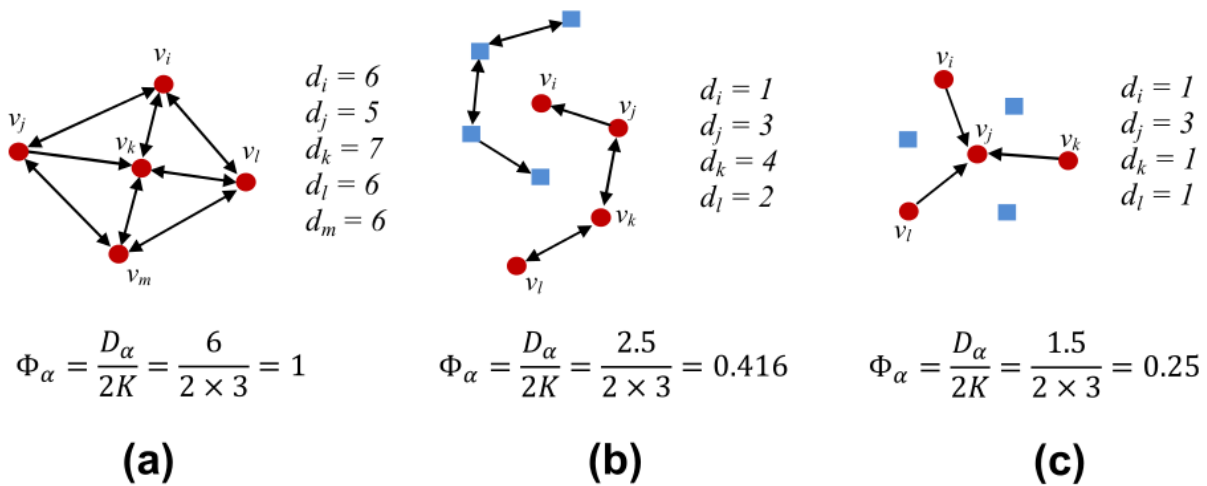


Figure 2.13: Calculation of the purity measure in 3-nearest neighbors networks. (a) A complete component where all vertices share the same label. (b) Two different classes grouped into two components with intermediary purity values. (c) A component surrounded by noise. Figure extracted from Bertini *et al.* (2011).

2.13 depicts three small networks to exemplify the purity measure: a complete component, two neighbor components and a component surrounded by noise.

The third and last step to construct the kAOG is by adapting each component to get its best purity for a specific k value. The rationale behind this operation is described in (Bertini *et al.*, 2011) as follows. A graph obtained by a unique value of k rarely 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, each component must have its own optimal k . As a result, all components reach their respective highest purity and form the kAOG. In the process for constructing the kAOG, k is increased 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 is used to compare among components of different k -associated graphs formed with different values of k . The component with the highest purity value is held, and the others are discarded.

With the kAOG at hand, a nonparametric Bayes classifier is used in order to predict unlabeled data. Since each component in the constructed graph contains vertices (cases) of the same class, it is possible to compute the probability of a new vertex to be classified to a given class by determining the probability of the same vertex to belong to each of the components. Specifically, the *a priori* probabilities are calculated by using a normalized purity value of each of the class components.

Development of network-based techniques for supervised learning

The techniques presented in this chapter deals with supervised classification problems. As it is described in section 2.1.1, in this situation there are available data labels which the classification process can rely on. The labeled set can be used to train and adjust the classifier which is further applied to classify unlabeled data. In the area of the network-based techniques, few efforts have been made in the supervised paradigm, since the majority of the techniques is unsupervised or semi-supervised. Here, we introduce four supervised techniques based on networks which have been developed during the doctorate period.

The first technique uses the random walk process to classify data. A new classification heuristic is introduced: instead of spreading labels or defining decision borders for the classifier, the labeling process is viewed as the ease of access unlabeled instances have to each available class. Basically, a set of labeled instances is mapped as the state space set for a random walker. The space set configures a network in which links represent transition probabilities. This network is further modified by a link weight composition which takes into account the bias information of the unlabeled instance to be classified. The limiting probabilities are calculated for the modified network structure resulting in label probabilities for the unlabeled instance. Due to the network representation, this technique is able to identify and distinguish classes presenting different shapes, and the random walk dynamical process is able to take both local and global label information into account during the classification process.

The second technique performs high-level data classification. Traditional supervised classification techniques define decision borders in the attribute space and the

unlabeled sample is associated to a class depending on its position in relation to these borders. Thus, the process is mainly based on the physical features of the data sets, and here we call this heuristic as low-level classification. In another way, a different classification heuristic is introduced in this section. It takes into account the networked structural information of the data sets. To be able to capture this structural information, the labeled data are mapped into underlying networks and specific network measures calculate modifications in network structure when an unlabeled vertex is inserted. Hence, the classification is performed by using both low-level (similarity measures) and high-level (structural measures) information.

The third technique deals with the problem of multiple observation sets. In this problem, a group of observations of a same pattern must be classified at once, that is, the whole group must receive the same class label. A labeled data set provides labeled observations of groups of patterns to train the classifier. The network-based scheme permits the detection of subnetworks which represent each group of observations, and the classification process takes into account the overall link strength between two subnetworks. The unlabeled subnetwork receives the label of the most related labeled subnetwork. The relationship strength is measured by two methods: modularity and Katz measures. The modularity measures how modular are two subnetworks when they are connected together: the most modular they are, the weaker is the label relationship between them. The Katz measure computes how central the nodes of an unlabeled subnetwork are in relation to a labeled network: the more central the nodes are, the stronger the label relationship between the subnetworks is.

The last technique presented in this section is an application of network-based methods to the problem of dimensionality reduction. Many applications in data mining, machine learning and pattern recognition face problems when computing similarities among data samples. When data lies in a high-dimensional space, these problems are often due to the “curse of dimensionality”. In this situation, similarity measures among data suffer from distortions, that is, when the dimensionality increases, the volume of the space increases so fast that the available data samples become sparse. Specifically, this situation is often found when dealing with images, which possess a large dimensional feature space, that is, images are high-dimensional patterns. One way to alleviate this problem is by performing dimensionality reduction, which aims at reducing the dimension of the input data in order to achieve a small set of features that keeps the most important original relationships among data samples. In the technique described in this chapter, the kAOG construction method is used to create networks that are embedded into a graph-embedding general network for dimensionality reduction. A new algorithm is proposed for the kAOG method to be able to create penalty networks required by the general framework. The technique is then compared to other network formation methods when using this same framework.

3.1 Measuring ease of access to data classes by using random walk

The general idea of the technique proposed by Cupertino and Zhao (2013a) is explained as follows. Random walk theory can be understood in terms of Markov chains. A Markov chain is formed by a sequence of states visited by a random walker, in which the probability to visit a given state is independent of past visits given that the current state is known (Gallager, 1995). The probability of moving from one state to another is called transition probability. It can be shown that, under some conditions and after an infinite number of transitions, the random walk process reaches a stationary state, also called limiting probabilities, which is independent of the initial state (Çınlar, 1975). In this situation, states that have larger incoming transition probabilities comparing to other states result in larger limiting probabilities, that is, the random walker has some preference to visit them. In other words, by representing the states as network nodes and the transition probabilities as link weights, we can say that nodes which are better linked to other nodes (have stronger link weights) result in larger limiting probabilities. In this case, the random walker prefers to visit some nodes in the network in detriment of others, that is, some nodes are easier accessed than others.

By following the previous reasoning, it is possible to classify unlabeled instances by using limiting probabilities. A set of labeled instances is mapped as network nodes, or as the state space set, that is, each node (labeled instance) is a possible state for the random walker. This network of labeled nodes is then modified by a specific link weight composition which takes into account the bias information of the unlabeled instance. The bias information changes the network structure by affecting the link weights among nodes resulting in a structure such that, after the calculation of the limiting probabilities, the most easily reached labeled nodes represent the class label of the unlabeled instance.

3.1.1 Technique description

The classification problem concerned within this section requires a given labeled data set, $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$. Each instance in this set has a single assigned label $l \in \mathcal{L}$. It is also given an unlabeled data set, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, containing instances that will be labeled after classification. Each instance is described by q attributes $\mathbf{x}_i^{(l)} = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$, and $\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$. The proposed technique can then be divided into two phases, training and classification, as it is described next.

Training phase:

In the training phase, a weighted and undirected network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ is constructed. Nodes represent data instances, $\mathcal{V} = \mathcal{X}^{(l)}$, and link weights represent similarities among instances, $\mathcal{E} = [\mathcal{W}_{ij}]$, $i, j = 1, \dots, n$. The similarity between any pair of instances $\mathbf{x}_i^{(l)}$ and $\mathbf{x}_j^{(l)}$ is denoted by w_{ij} . The network similarity matrix $\mathcal{W} = \{w_{ij}\}$ can be calculated by using any distance function. Specifically, we use the Euclidean distance in all experiments in this section. The resulting network is called training network.

Classification phase:

To classify an unlabeled instance $\mathbf{x}^{(u)}$, a vector $\mathcal{S} = [s_1, s_2, \dots, s_n]$ containing the link weights between $\mathbf{x}^{(u)}$ and all other nodes $\mathbf{x}_i^{(l)}$ is calculated. The link biases of node $\mathbf{x}^{(u)}$ are inserted into the training network \mathcal{N} by calculating its link weights to all other nodes into this network. Then, an asymmetric and $n \times n$ modified similarity matrix $\hat{\mathcal{W}}$ is constructed by the following composition:

$$\hat{\mathcal{W}} = \mathcal{W} + \epsilon \hat{\mathcal{S}}, \quad (3.1)$$

where ϵ is a non-negative parameter and $\hat{\mathcal{S}}$ is the following $n \times n$ matrix:

$$\hat{\mathcal{S}} = \begin{bmatrix} \mathcal{S}^{(1)} \\ \mathcal{S}^{(2)} \\ \vdots \\ \mathcal{S}^{(n)} \end{bmatrix}.$$

Remark 1. It can be observed in Eq. 3.1 that the link weight biases of the unlabeled instance $\mathbf{x}^{(u)}$, encoded in matrix $\hat{\mathcal{S}}$, are applied over all links \mathcal{W} of the training network \mathcal{N} , that is, the weight of each link is linearly added up with its corresponding weight bias. The idea behind this operation is that the distance between any pair of nodes is modified due to the new network routes introduced by the insertion of the weight biases of the unlabeled instance. The larger is the similarity between the unlabeled instance and a node, say node i , the more strengthened the connections from all other nodes to node i are after this operation. The parameter ϵ controls the influence of the weight biases. The larger is the value of parameter ϵ , the larger is the influence of the weight bias of $\mathbf{x}^{(u)}$.

The steps described above can be easily understood by using the toy example depicted in Fig. 3.1. In this example, a training network is formed by 4 labeled instances belonging to 2 distinct classes, *green disks* and *orange squares*, each one containing 2 representative instances, $\{1, 2\}$ and $\{3, 4\}$, respectively (Fig. 3.1a). In this initial training

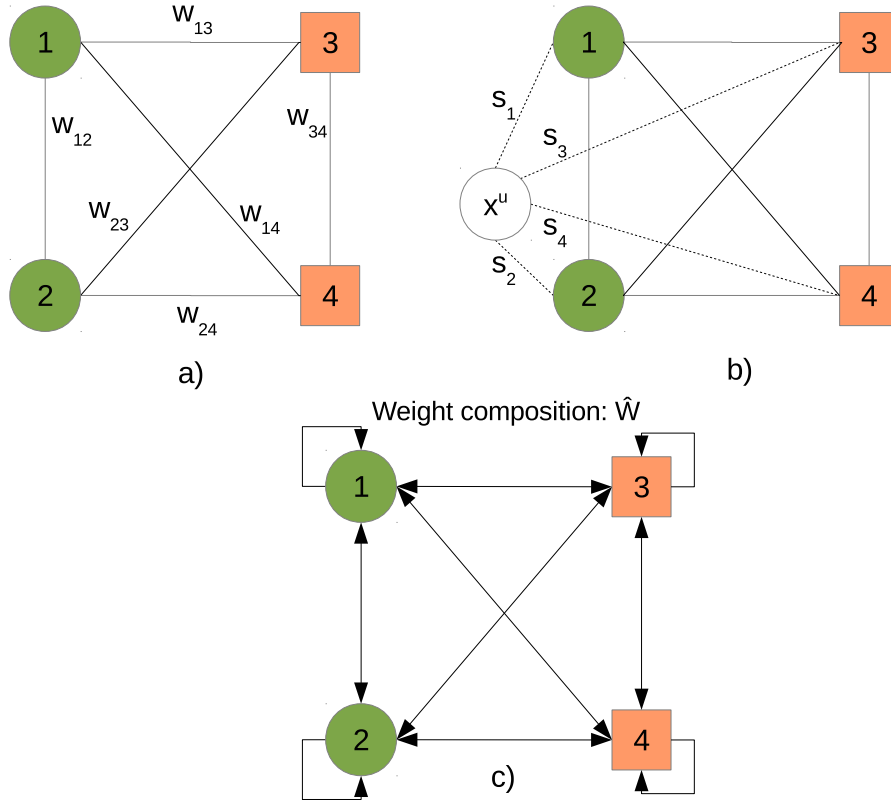


Figure 3.1: Illustration to calculate the modified connection matrix \hat{W} when classifying the unlabeled instance x^u . a) An undirected network \mathcal{N} is formed by using 4 training instances of 2 classes: green and yellow; b) the vector of similarities S are calculated between x^u and all training nodes; c) modified network \mathcal{N} , directed with self-loops, after the bias composition (Eq. 3.1).

network, the links are undirected and the similarity matrix \mathcal{W} is symmetric. Given an unlabeled instance x^u , the similarity vector S between x^u and all other nodes is computed (Fig. 3.1b). After this computation, the link weight biases of x^u are added up to the original similarity matrix to form a biased similarity matrix for the same network (Eq. 3.1). It can be seen from Fig. 3.1c that the network becomes directed, with self-loops, and the biased weight matrix is no more symmetric.

After the preceding operations, we have at hand a biased similarity matrix \hat{W} or, in other words, the adjacency matrix of the modified training network \mathcal{N} , which is called classification network. By using this network, it is now possible to apply the random walk limiting probabilities through the states represented by the network nodes. The transition probabilities can be found by means of the matrix \hat{W} . To compute the entries of the transition matrix $\mathcal{P} = [\mathcal{P}_{ij}]$, the entries of matrix \hat{W} are normalized:

$$p_{ij} = \hat{w}_{ij} / \sum_{j=1}^n \hat{w}_{ij}.$$

With the above matrix \mathcal{P} at hand, the limiting probabilities can be calculated by finding the eigenvector corresponding to the unit eigenvalue of matrix \mathcal{P}^T , the transpose of matrix \mathcal{P} , or by iterating the system

$$\mathbf{p}_{i+1} = \mathcal{P}^T \mathbf{p}_i \quad (3.2)$$

to the stationary state, where \mathbf{p} is an n -dimensional normalized vector. The limiting probabilities result in the form of the following vector:

$$\mathbf{p}^\infty = [p_1 \ p_2 \ \dots \ p_n],$$

where each element represents a state, and each entry p_i can be interpreted as the probability that $\mathbf{x}^{(u)}$ belongs to the class of state i . As the final step, the classification of $\mathbf{x}^{(u)}$ is accomplished by assigning it the most representative label from the set of states. A set \mathcal{T} containing the τ states with the largest limiting probabilities are selected and the most representative class in \mathcal{T} is associated to $\mathbf{x}^{(u)}$.

3.1.2 Algorithm and complexity analysis

In a concise form, the proposed supervised inductive classification technique can be summarized by Algorithm 3.

Algorithm 3 Network-based classification by random walk ease of access

Input:

$\mathcal{X}^{(l)}$: training data set

$\mathbf{x}^{(u)}$: unlabeled instance

Parameters:

ϵ : weighting for bias link composition

τ : number of largest probabilities

Output:

l : estimated label for $\mathbf{x}^{(u)}$ ($l \in \mathcal{L}$)

Training:

1 : \mathcal{N} = Create a training network from $\mathcal{X}^{(l)}$

Classification:

2 : $\hat{\mathcal{W}}$ = Compose bias link weights into \mathcal{N} (Eq. 3.1)

3 : \mathbf{p}^∞ = Compute limiting probabilities (Eq. 3.2)

4 : \mathcal{T} = Select the τ largest limiting probabilities from \mathbf{p}^∞

5 : l = Assign $\mathbf{x}^{(u)}$ the most representative label in \mathcal{T}

The computational complexity of the proposed technique is analyzed considering the steps from 1 through 4 of Alg. 3. The creation of a training network in step 1 requires a computation of $O(n^2)$ since the similarities between all pair of instances is calculated. The weight biases composition needs $O(n^2)$ operations since each matrix

entry must be added in step 2. The limiting probabilities in step 3 can be promptly calculated by iterating the system of Eq. 3.2 to the stationary state in $O(n^2)$ computations. Step 4 of Alg. 3 is $O(n \log n)$ by using an efficient sorting algorithm such as the Quicksort algorithm (Cormen *et al.*, 2003). Putting it all together, the computational complexity of the proposed technique is $O(n^2 + n^2 + n^2 + n \log n)$. Taking the highest order term, it results in $O(n^2)$.

Nevertheless, this complexity can be reduced by dealing with sparse networks such as the k -nearest neighbor networks, in which connection matrices are sparse (all networks constructed in the simulations of this section fall into this category). In this case, the complexity of steps 2 and 3 are reduced to $O(n \langle k \rangle)$, being $\langle k \rangle (\langle k \rangle \ll n)$ the average degree of the network (average number of links). In addition, by using the graph construction method based on Lanczos bisection (Chen *et al.*, 2009b), step 1 requires $O(n^t)$, and the technique complexity reduces to $O(n^t + n \langle d_g \rangle + n \langle k \rangle + n \log n)$. Furthermore, according to (Chen *et al.*, 2009b), a small value for parameter t ($1.06 \leq t \leq 1.33$) is sufficient to achieve high quality networks. Thus, the computational complexity order of the proposed technique lies in $O(n^{1.06})$ to $O(n^{1.33})$.

The complexity comparison with the optimized implementations of the simulated techniques shown next (section 3.1.3) is as follows: i) kAOG requires from $O(n^{1.06})$ to $O(n^{1.33})$ computations (Bertini *et al.*, 2011); ii) standard k -NN-based techniques requires $O(n)$ (it can be $O(n^{0.5})$ when optimized with kd tree methods (Grother *et al.*, 1997)); iii) for C4.5, tree induction requires $O(n(\log n)^2)$ (Quinlan, 1992) ($O(cn)$ in specific cases (Su and Zhang, 2006)); iv) finally, standard SVM is $O(n^3)$, and state-of-the-art implementations have empirically a training time that scales between $O(n)$ and $O(n^{2.3})$ (Platt, 1999; Tsang *et al.*, 2005).

3.1.3 Experimental results

In this section, we show some numerical results of the proposed supervised technique as well as a comparative study with some well-known classification techniques. In the experiments, 15 data sets were selected from the UCI machine learning repository (Bache and Lichman, 2013), and a large data set was selected from the MNIST database of manuscript digits (LeCun *et al.*, 1998). Table 3.1 shows the metadata for all data sets. As can be seen in this table, the selection was made to encompass diversity on data domains as well as to consider different number of classes (from 3 to 15), attributes (from 4 to 784) and data set sizes (from 101 to 10,000). The Euclidean distance was used in all simulations as the distance measurement. Eventual categorical attributes, in data sets such as Balance and Zoo, were treated as numerical. As a data preparation, each attribute vector was normalized to have a magnitude of 1. Individual cases were normalized by dividing each attribute of an instance by the square root of the sum of

Table 3.1: Metadata of data sets used in simulations.

Domain	Instances	Attributes	Classes
Zoo	101	16	7
Hayes-Hoth	132	5	3
Iris	150	4	3
Teaching	151	5	3
Wine	178	13	3
Image	210	19	7
Glass	214	9	6
E. Coli	336	8	8
Libras	360	91	15
Balance	625	4	3
Vehicle	846	18	4
Vowel	990	13	11
Yeast	1,484	8	10
Wine Q. (Red)	1,599	12	6
Segment	2,310	19	7
MNIST	10,000	784	10

the squares of the individual attributes. Thus, an instance $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iq})$ was normalized by dividing each attribute x_{ij} by $\sum_{j=1}^q x_{ij}^2$.

The parameter optimization for the proposed technique was done as follows. Training networks \mathcal{N} (step 1 of Alg. 3) were created by using the Euclidean measurement as a distance function to determine the initial link weights between neighbor nodes. Parameter τ (Step 4 of Alg. 3) ranged from 1 to the number of instances in the largest class of the training set. Parameter ϵ (Eq. 3.1) was evaluated by using the grid method in the interval $\{0, 0.1, 0.2, \dots, 10\}$.

As it is stated in section 3.1.1, ϵ is responsible for weighting the biases provided by the unlabeled instance to the training network. Figure 3.2 depicts the classification accuracy in function of parameter ϵ for some simulated data sets. The results were averaged over 50 runs. Each run was performed by using a 10-fold stratified cross-validation process (Kim, 2009). In this process, the data set is split into 10 disjoint sets and, in each run, 9 sets are used as the training set and 1 set is used as the test data, resulting in a total of 10 runs. Therefore, 50×10 runs were executed. It can be seen on Fig. 3.2 that next to value 0 - where the link bias influence is very reduced - the classification accuracies are poor. On the other hand, as ϵ becomes larger, the accuracies increase and stabilize. In this case, the link biases of the unlabeled instance play a main role due to the large weight applied to them (Eq. 3.1). These scenarios configure a convergent behavior for parameter ϵ and can help in the experiments by restricting the search space.

The proposed technique was compared to other 6 well-known and established classification algorithms: k -Nearest Neighbors (kNN) (Tan *et al.*, 2005), Weighted

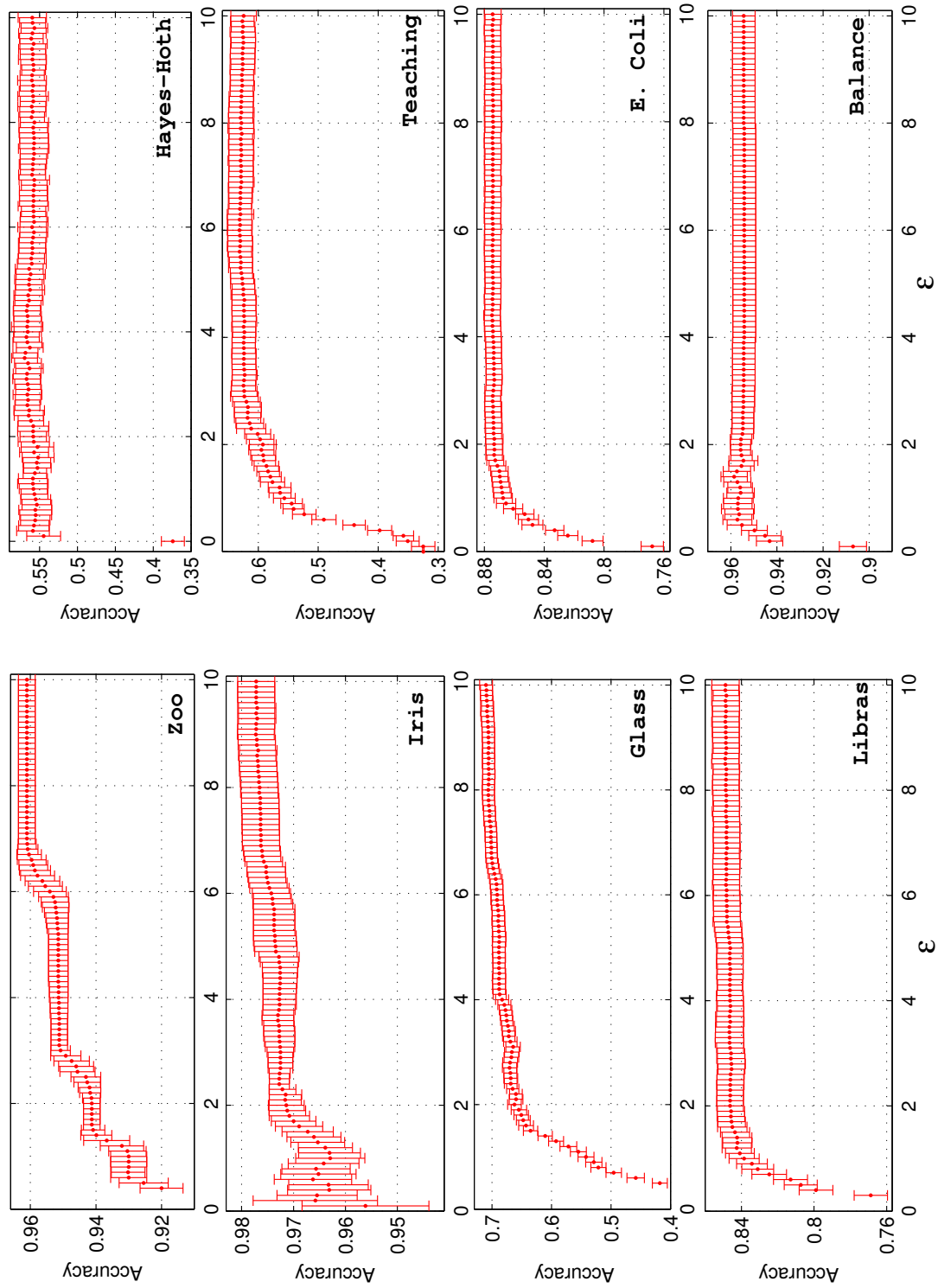


Figure 3.2: Classification accuracy in function of parameter ϵ for the simulated data sets. The curves show standard deviations for each point. 50 simulations were averaged.

kNN (WkNN), Prototype-based kNN (kNN) (Hastie *et al.*, 2009), Decision Tree C4.5 (Quinlan, 1992), Multi-Class SVM (MSVM) (Vapnik, 1999) and the network-based k-Associated Optimal Graph (kAOG) (Bertini *et al.*, 2011). For the parametric algorithms (all algorithms except kAOG), a repeated cross-validation (Kim, 2009) was done in order to optimize their respective parameters. For the MSVM algorithm, we used the *one-against-one* multi-class version, in which $\mathcal{L}(\mathcal{L} - 1)/2$ binary classifiers distinguishes between every pair of classes by using a voting scheme. To avoid ties, the output of each MSVM corresponded to the real valued decision functions. For reducing the parameter search space in MSVM model selection, the only kernel in consideration was the radial basis function, $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, and the stopping criterion for the optimization method was defined as the Karush-Kuhn-Tucker violation to be less than 10^{-3} , the same condition used in (Hsu and Lin, 2002). For kNN algorithms, the only parameter is the number of neighbors k , which ranged from 1 to the number of instances in the largest class of the training set. In the WkNN technique, the classification process was performed by using the sum of the weights between the instance to be labeled and its k -nearest neighbors. Specifically, the weight between two instances \mathbf{x}_i and \mathbf{x}_j was defined as $1/\|\mathbf{x}_i, \mathbf{x}_j\|$, where $\|\cdot\|$ is the Euclidean distance. For the C4.5 algorithm, two parameters were adjusted: the confidence factor that took the values $cf \in \{0, 0.1, 0.25, 0.5, 0.8, 1\}$, where smaller values incur more pruning (1 is for no pruning), and the minimum number of instances that a set must have in order to be further partitioned, $m \in \{0, 1, 2, 3, 4, 5, 10, 15, 20, 50\}$.

Table 3.2 shows the classification accuracy on the test set followed by the standard deviation, and Table 3.3 shows the rank of the techniques for each data set. The calculation procedure for the rank measurement is as follows: i) for each data set, the algorithms were ranked according to their average classification accuracy, that is, the best algorithm was ranked as first, the second best was ranked as second, and so on; and ii) for each algorithm, the average rank was based on the rank values on all the data sets. It can be seen that the proposed technique achieved the best average ranking among all simulated techniques.

To complete the experimental analysis, a statistical test proposed in (Demšar, 2006) was used to compare the proposed technique to the other techniques over the multiple data sets. According to the statistical test, the average rank of each technique is used in the non-parametric Friedman test to check whether they are significantly different from the average value of the overall ranks (3.91 from the results in Table 3.3). The Friedman test uses the F -distribution to calculate a critical value in order to reject the null-hypothesis (that is to say, all techniques are equivalent) under a defined confidence level. Being N_a the number of tested techniques and N_b the number of simulated data sets, the critical value is determined by the degrees of freedom ($N_a - 1$) and $(N_a - 1) \times (N_b - 1)$. In our simulations, $N_a = 7$ and $N_b = 16$, and so the de-

Table 3.2: Classification accuracy (%) followed by standard deviation. Each result shows the adjusted parameters for model selection. Best results for each data set are in bold face.

Data set	Proposed (ϵ, τ)	kAOG	kNN (k)	WkNN (k)	PkNN (p)	C4.5 (cf, m)	MSVM (cp, γ)
Zoo	96.2 ± 0.1 (7.8, 1)	97.0 ± 5.2	96.1 ± 5.9 (1)	96.2 ± 5.8 (1)	93.6 ± 7.1 (2)	95.8 ± 5.3 (0.1, 0)	96.3 ± 6.4 (2 ¹ , 2 ¹)
Hayes-Hoth	57.0 ± 2.3 (3.5, 1)	55.7 ± 12.6	54.9 ± 12.4 (1)	56.8 ± 13.2 (3)	44.6 ± 11.3 (8)	46.7 ± 10.5 (1, 0)	45.4 ± 13.1 (2 ¹² , 2 ¹³)
Iris	98.1 ± 1.1 (3.1, 29)	97.4 ± 3.2	97.9 ± 3.4 (19)	97.9 ± 3.3 (19)	97.3 ± 3.2 (3)	95.0 ± 5.8 (0.25, 2)	97.0 ± 4.6 (2 ⁻² , 2 ³)
Teaching	63.1 ± 2.0 (6.0, 1)	62.5 ± 11.6	59.6 ± 10.2 (1)	63.0 ± 12.3 (9)	58.3 ± 9.0 (29)	58.2 ± 14.9 (1, 0)	52.5 ± 7.9 (2 ⁶ , 2 ³)
Wine	84.6 ± 1.6 (9.4, 1)	83.5 ± 8.5	84.1 ± 8.5 (1)	84.1 ± 8.2 (1)	81.4 ± 11.9 (28)	91.7 ± 6.7 (0.5, 1)	94.4 ± 5.8 (2 ¹¹ , 2 ²)
Image	75.5 ± 0.8 (1.8, 1)	75.3 ± 7.5	75.3 ± 8.2 (1)	75.4 ± 8.2 (3)	75.5 ± 10.2 (16)	80.7 ± 7.6 (0.8, 3)	86.7 ± 7.4 (2 ¹⁰ , 2 ⁻³)
Glass	72.5 ± 1.1 (10, 2)	72.5 ± 8.1	71.9 ± 8.6 (1)	71.8 ± 9.0 (1)	67.3 ± 11.8 (6)	66.9 ± 9.4 (0.1, 3)	69.5 ± 5.6 (2 ¹⁰ , 2 ⁴)
E. Coli	87.5 ± 0.6 (4.4, 9)	85.8 ± 6.4	86.5 ± 5.2 (9)	87.4 ± 5.4 (9)	80.4 ± 5.2 (2)	83.6 ± 6.1 (0.25, 3)	86.7 ± 8.2 (2 ¹² , 2 ⁻⁹)
Libras	84.9 ± 0.8 (9.0, 1)	85.4 ± 5.3	84.8 ± 5.5 (1)	84.8 ± 5.4 (1)	55.7 ± 4.2 (13)	71.6 ± 7.5 (0.8, 1)	86.6 ± 5.0 (2 ⁷ , 2 ²)
Balance	96.3 ± 0.6 (4.1, 9)	94.9 ± 2.5	94.7 ± 2.6 (1)	96.7 ± 2.1 (11)	76.6 ± 5.7 (8)	89.6 ± 3.7 (0.5, 1)	98.2 ± 0.9 (2 ⁷ , 2 ⁰)
Vehicle	67.6 ± 0.6 (10.0, 6)	67.9 ± 4.4	67.3 ± 4.0 (3)	67.6 ± 4.1 (5)	60.0 ± 5.5 (10)	70.7 ± 3.5 (0.5, 2)	84.4 ± 3.4 (2 ¹⁰ , 2 ³)
Vowel	97.7 ± 0.3 (2.8, 1)	98.9 ± 0.7	97.8 ± 1.0 (1)	98.8 ± 0.9 (11)	96.6 ± 1.8 (8)	78.6 ± 4.3 (0.5, 0)	97.5 ± 1.9 (2 ⁷ , 2 ⁰)
Yeast	59.5 ± 0.6 (9.6, 13)	53.6 ± 3.8	58.7 ± 3.4 (15)	60.9 ± 3.6 (16)	48.0 ± 2.7 (1)	55.8 ± 3.6 (0.1, 5)	58.9 ± 4.8 (2 ¹¹ , 2 ⁰)
Wine Q. Red	62.3 ± 0.5 (6.4, 1)	61.8 ± 3.6	61.3 ± 3.4 (1)	64.0 ± 3.8 (19)	38.9 ± 3.1 (27)	59.8 ± 2.4 (1, 0)	60.4 ± 3.2 (2 ⁹ , 2 ¹)
Segment	93.7 ± 0.2 (9.2, 1)	93.7 ± 1.5	93.6 ± 1.6 (1)	93.6 ± 1.4 (5)	60.9 ± 3.2 (25)	95.4 ± 1.2 (1, 0)	96.6 ± 1.2 (2 ¹¹ , 2 ⁰)
MNIST	96.3 ± 0.1 (10.0, 4)	95.1 ± 1.7	95.3 ± 2.3 (1)	95.7 ± 1.1 (5)	94.3 ± 2.2 (25)	94.5 ± 1.2 (1, 0)	96.4 ± 1.3 (2 ¹¹ , 2 ⁰)

Table 3.3: Rank of algorithms followed by standard deviation for the results in Table 3.2. The best ranks are in boldface.

Data set	Proposed (ϵ, τ)	kAOG	kNN (k)	WkNN (k)	PkNN (p)	C4.5 (cf, m)	MSVM (cp, γ)
Zoo	3	1	5	3	7	6	2
Hayes-Hoth	1	3	4	2	7	6	6
Iris	1	4	2	2	5	7	6
Teaching	1	3	4	2	5	6	7
Wine	3	6	4	4	7	2	1
Image	3	6	6	5	3	2	1
Glass	1	1	3	4	6	7	5
E. Coli	1	5	4	2	7	6	3
Libras	3	2	4	4	7	6	1
Balance	3	4	5	2	7	6	1
Vehicle	4	3	6	4	7	2	1
Vowel	4	1	3	2	6	7	5
Yeast	2	6	4	1	7	5	3
Wine Q. Red	2	3	4	1	7	6	5
Segment	3	3	5	5	7	2	1
MNIST	2	5	4	3	7	6	1
Average rank	2.31 \pm 1.08	3.50 \pm 1.75	4.19 \pm 1.05	2.88 \pm 1.31	6.38 \pm 1.15	5.06 \pm 1.91	3.06 \pm 2.31

degrees of freedom are 6 and 90. For a significance level of 5%, the critical value for the experiments is $F(6, 90) \approx 2.20$. So, any result of the Friedman test larger than this critical value rejects the null-hypothesis. For the presented numerical results, the null-hypothesis was rejected because the Friedman test resulted in $F_F \approx 5.04$ and, therefore, the simulated techniques are not equivalent.

In the next step, the pairwise accuracy comparison of our technique in relation to the others is done following the Bonferroni-Dunn test (Demšar, 2006; Dunn, 1961). This test consists in comparing a control technique against the others. Here, the control technique is the proposed technique. A critical difference measure (CD) is calculated for a given critical value (q_β) for a significance level β . Any average rank difference larger than CD confirms that two techniques are significantly different. Thus, by using a significance level of $\beta = 0.05$, the critical value is $q_{0.05} = 2.64$, resulting in $CD = 2.01$. So, any ranking difference larger than 2.01 justifies the superiority of the best ranked algorithm. Using pairwise comparisons in Table 3.3, we see that our technique is better than PkNN and C4.5 (rank differences of 4.07 and 2.75 respectively) with a significance level of 5%, and presents no significant difference to kNN, WkNN, MSVM and kAOG. Thus, we can conclude that the proposed technique is at least comparable to the simulated techniques. It should also be noted that in the comparisons all techniques have been optimized with respect to their parameters for model selection.

3.2 High-level inductive classification

Due to the importance of the supervised learning paradigm in various real applications, many classification techniques have been developed, such as the k -Nearest Neighbors (k NN) (Tan *et al.*, 2005), Linear Discriminant Analysis (LDA) (Duda *et al.*, 2000), Naive-Bayes (Friedman *et al.*, 1997), Neural Networks (Haykin, 1998), Support Vector Machines (SVM) (Burges, 1998), Decision Trees (Quinlan, 1992) and so on. Although each of them has its own features, all these traditional classification techniques share the same heuristic: the trained classifier defines decision borders in the data space and the label induction phase verifies the relative position of each unlabeled instance to these borders. Therefore, the data space, usually the Euclidean space, is divided into subspaces, each one representing a data class. These subspaces are not overlapped in the case of crisp classification, but they can be slightly overlapped in the case of fuzzy classification. In either way, strong twisting or largely overlapped subspaces are not permitted. In other words, traditional classification divides the data space according to physical features (similarity, distance, or distribution) of the training data, ignoring many other intrinsic and semantic relations among data items, which usually generate complex shaped classes in the data space. On the other hand, it is well known that the human (animal) brain is able to identify patterns according to semantic mean-

ings of the input data. Thus, it can be useful to perform data classification beyond the usual **data space division** concept. In this context, network-based techniques can provide new contributions to this research area by performing data classification from quite different viewpoints other than the traditional data space division way.

For instance, the authors in (Silva and Zhao, 2012c) propose a method to classify unlabeled instances by means of static network structural measures. In their technique, an unlabeled instance receives the label from the data network of which structure is kept unmodified or is barely modified after the insertion of the unlabeled instance. This approach permits one to understand a class as a data pattern, and the classification process is conducted by checking the **pattern conformation**, that is, a data item is classified into a given class if it confirms the pattern formed by that class no matter how far it is from the class centroid. This method can generate complex shaped data subspaces and reveal some semantic meaning of the training data. Alternatively, in (Cupertino *et al.*, 2012) (see subsection 3.3.2), we consider that unlabeled instances belong to a sub-network (class) which results in the lowest modularity value (Newman and Girvan, 2004) after connecting them to a network constructed from the unlabeled set. In that approach, the classification process is performed by considering the **connectivity pattern** of the training data. In this case, low modularity values mean that a network is well connected, that is, node instances are strongly related to each other and likely belong to the same class. In another recent study (Cupertino and Zhao, 2012b) (see subsection 3.3.3), we propose the use of node centrality for data classification. Their technique is capable of classifying multiple observations where each pattern is represented by a group of invariant transformations. The classifier must predict the pattern this group belongs to. In this approach, the classification is conducted by analyzing **how central** or **how important** a test instance is to each class. Instead of classifying a data instance by similarity or distribution, as it is done in traditional techniques, the test instance receives the label from the class where it acts as an important piece. Hence, the classification is made by using the heuristic of **node importance**.

Following this basis of reasoning, we introduced a new network-based classification technique which considers the **ease of access** heuristic and **network structure** measurements to perform high-level classification (Cupertino and Zhao, 2013c). The proposed technique uses a measure for the dynamical process called random walk limiting probabilities through an underlying network that combines both data similarity and structural measures. The training data set is used to construct the network, in which instances (nodes) represent the states a random walker visits during the random walk process. An instance is considered to belong to the class most easily reached by the random walker, that is, the limiting transition probability of a random walker to that class is the largest. In this way, both local and global node relationships are taken into account. Moreover, the proposed approach is actually a general classifi-

cation scheme in such a way that one can put any new classification criteria in the weight matrix to guide the random walk process. We show how the low level classification term, represented by similarities between data items, and the high level term, represented by a network structural measure, are combined in this scheme. As a consequence, the proposed approach can classify data not only using physical features, but also checking the structural pattern formation via the network constructed from the training data. We also show that such a combination can improve the classification results in real applications, such as image recognition. Another interesting finding of this work is that the high level term embedded in the connection matrix of the data network is specially useful in complex situations where the low level term fails such as when there is a high mixture of data classes.

3.2.1 The general classification framework

As a general framework, the limiting probabilities are calculated through the following modified connection weight matrix $\hat{\mathcal{W}}$:

$$\hat{\mathcal{W}} = \alpha \mathcal{W}_{sim} + (1 - \alpha) \mathcal{W}_{str}, \quad (3.3)$$

where \mathcal{W}_{sim} is the similarity connection weight matrix, \mathcal{W}_{str} is the structural connection weight matrix, and $\alpha \in [0, 1]$ is a tuning parameter of the convex combination.

Each term of Eq. 3.3 has a distinct and specific role. The similarity weight matrix (\mathcal{W}_{sim}) takes into account the similarities among instances in the attribute space, that is, each element w_{ij} in \mathcal{W}_{sim} represents the similarity between data samples \mathbf{x}_i and \mathbf{x}_j . By adopting the cluster assumption, instances lying close to each other in the attribute space should belong to the same class. So, this term provides similarity bias values to the classification process. In other words, this matrix considers the physical features of the training data and it alone leads to the low-level classification. Although any distance function could be applied to measure the similarity between data samples, the Euclidean distance is used in all simulations presented in this section.

On the other hand, the structural weight matrix (\mathcal{W}_{str}) also provides a classification bias, but considering information from a different nature: it complements the similarity weight matrix by providing structural information, which is a measure of how the network structure of each class is affected after the insertion of an unlabeled node. By adopting this scheme, an instance should belong to the network presenting the smallest structural change, that is, the new inserted node keeps the conformity of that network. This matrix considers the pattern formation of the training data, providing high-level classification results.

Furthermore, as a general framework, Eq. 3.3 does not fix how the similarity weight matrix should be composed neither which structural network measure or a combina-

tion of measures should be taken into account. Actually, both terms can be arbitrarily defined by considering the classification interests. Our proposed choices for this framework are explained in details in the next subsections.

The classification problem concerned within this section requires a labeled data set, $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, where each instance has a single assigned label $l \in \mathcal{L}$. It is also given an unlabeled data set, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, containing instances that will be assigned to labels after classification. Each sample is described by q attributes, $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$, and $\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$.

The classification process can be divided into the phases of training and classification as follows:

Training phase

The training phase consists in creating a weighted and undirected network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ without self-loops. In this network, nodes represent labeled data instances in the set $\mathcal{V} = \mathcal{X}^{(l)}$, and similarities among instances are represented by edge weights in $\mathcal{E} = \{w_{ij}\}$, $i, j = 1, \dots, n$. The similarity matrix $\mathcal{W} = [w_{ij}]$ is calculated by some sort of distance function (e.g., Euclidean), and each entry w_{ij} is the similarity between a pair of instances $\mathbf{x}_i^{(l)}$ and $\mathbf{x}_j^{(l)}$. In this step, a single network is generated using all data samples in the training set, that is, data samples from different classes are mixed into a single network.

Classification phase

To classify an unlabeled instance $\mathbf{x}^{(u)}$, the limiting probabilities of the random walk process are calculated through matrix $\hat{\mathcal{W}}$ (Eq. 3.3). These probabilities are found by means of a transition matrix \mathcal{P} of which entries correspond to the scaled entries of $\hat{\mathcal{W}}$:

$$p_{ij} = \hat{w}_{ij} / \sum_{j=1}^n \hat{w}_{ij}.$$

With the above matrix \mathcal{P} at hand, the limiting probabilities can be calculated by two ways: by finding the eigenvector corresponding to the unit eigenvalue of matrix \mathcal{P}^T , the transpose of matrix \mathcal{P} , or by iterating the following system:

$$\mathbf{p}_{i+1} = \mathcal{P}^T \mathbf{p}_i, \quad (3.4)$$

to the stationary state, where \mathbf{p} is an n -dimensional normalized vector. The iterating way is faster and results in a vector \mathbf{p}^∞ of which entry \mathbf{p}_i^∞ represents the probability $\mathbf{x}^{(u)}$ belongs to the class of node i . As the final step, a set \mathcal{T} comprising the τ -largest node probabilities is constructed, and $\mathbf{x}^{(u)}$ receives the label from the most representative

class in \mathcal{T} .

The construction of the modified connection weight matrix $\hat{\mathcal{W}}$ (Eq. 3.3) is detailed in the next subsections.

3.2.1.1 Similarity weight matrix

The similarity weight matrix provides a bias to the random walk process in the classification phase based on similarities among data. It conveys physical information of an unlabeled instance $\mathbf{x}^{(u)}$ in relation to the labeled instances in the training set. Similarly to Eq. 3.1 with $\epsilon = 1$ (see subsection 3.1.1), the similarities s_i computed between $\mathbf{x}^{(u)}$ and each training instance $\mathbf{x}_i^{(l)}$ are calculated to form a vector $\mathcal{S} = [s_1, s_2, \dots, s_n]$, and \mathcal{W}_{sim} is constructed by the following operation:

$$\mathcal{W}_{sim} = \mathcal{W} + \hat{\mathcal{S}}, \quad (3.5)$$

where \mathcal{W} is the similarity matrix resulted from the training phase and $\hat{\mathcal{S}}$ is the following $n \times n$ matrix:

$$\hat{\mathcal{S}} = \begin{bmatrix} \mathcal{S}^{(1)} \\ \mathcal{S}^{(2)} \\ \vdots \\ \mathcal{S}^{(n)} \end{bmatrix}.$$

For a detailed explanation of these operations, see Remark 1 and Fig. 3.1 in section 3.1.1.

3.2.1.2 Structural weight matrix

The structural term provides a bias to the classification process by taking into account the network structural information. It computes the change that a structural measure suffers after inserting node $\mathbf{x}^{(u)}$ into an initial training network. To compute this variation, a network \mathcal{G}_l is constructed for each class separately (resulting in $|\mathcal{L}|$ connection weight matrices, \mathcal{W}_l , where $|\mathcal{L}|$ is the number of available labels), and a structural measure $m(\cdot)$ is calculated for all nodes of each network \mathcal{G}_l separately:

$$\mathcal{V}_l = m(\mathcal{W}_l), \quad (3.6)$$

where \mathcal{V}_l is an n_l -dimensional vector, being n_l the number of instances (nodes) in the class (network) \mathcal{G}_l .

The unlabeled instance $\mathbf{x}^{(u)}$ is then inserted into each network \mathcal{G}_l . Again, the same measure $m(\cdot)$ is computed for these new networks (Eq. 3.6), resulting in n_l -dimensional vectors \mathcal{V}'_l (disregarding the measure for the unlabeled node that would

result in a $n_l + 1$ dimension). As large networks are more robust to structural changes, both measures \mathcal{V}_l and \mathcal{V}'_l are multiplied by their respective size factors n_l/n , and so a small structural change is counterbalanced by a large network size.

Finally, the structural measure vectors \mathcal{V}_l and \mathcal{V}'_l are grouped into two normalized vectors $\mathcal{V} = [\mathcal{V}_1^T, \dots, \mathcal{V}_L^T]$ and $\mathcal{V}' = [\mathcal{V}'_1^T, \dots, \mathcal{V}'_L^T]$, and the structural variations for all labeled nodes, before and after the insertion of $\mathbf{x}^{(u)}$, are calculated by the following absolute difference:

$$\Delta\mathcal{V} = |\mathcal{V} - \mathcal{V}'|.$$

The proposed network structure heuristic states that $\mathbf{x}^{(u)}$ should belong to the network presenting the smallest structural change. So, we compose the $n \times n$ matrix \mathcal{W}_{str} by the following operation:

$$\mathcal{W}_{str} = \begin{bmatrix} \Delta\mathcal{V}_{comp}^{(1)} \\ \Delta\mathcal{V}_{comp}^{(2)} \\ \vdots \\ \Delta\mathcal{V}_{comp}^{(n)} \end{bmatrix}, \quad (3.7)$$

where $\Delta\mathcal{V}_{comp}$ is the complementary vector: $\Delta\mathcal{V}_{comp} = 1 - \Delta\mathcal{V}$.

Remark 2. Eq. 3.7 encompasses the structural change regarding the network measure $m(\cdot)$ on each labeled node. For example, if node i suffers a large change on its structural measure after the insertion of node $\mathbf{x}^{(u)}$ into the network of node i , then $\Delta\mathcal{V}(i)$ will be large, but the complementary $\Delta\mathcal{V}_{comp}(i)$ will be small. As a consequence of the composition given by Eq. 3.3, each edge on $\hat{\mathcal{W}}$ that links to node i will receive a small bias value, and so the influence on the random walker will be small. Conversely, if node i suffers a small structural change after the insertion of $\mathbf{x}^{(u)}$, then $\Delta\mathcal{V}(i)$ will be small, but the complementary $\Delta\mathcal{V}_{comp}(i)$ will be large. As a consequence, each edge on $\hat{\mathcal{W}}$ that links to node i will receive a large bias value (according to Eq. 3.3), and the influence on the limiting probabilities will be large, guiding the random walker in direction of node i .

Many network measures can be used to characterize the structural information of the input data. For the sake of application, in this study we used the clustering coefficient (Watts and Strogatz, 1998) as the local structural measure $m(\cdot)$ in Eq. 3.6. This network measure is described in section 2.2.2. It is a convenient measure to be used within the general framework of Eq. 3.3 due to the following reason: while the similarity term \mathcal{W}_{sim} in Eq. 3.5 provides a global measure, taking into account the similarities among all nodes and configuring the whole network topology, the clustering coefficient acts as a complementary structural measure by computing the local information for each node.

3.2.2 Algorithm and complexity analysis

In a concise form, the proposed supervised classification process can be summarized by Algorithm 4. Steps 2 and 3 are responsible for computing the similarity and the structural matrix terms, respectively, which compose the general framework in Eq. 3.3. As have been described in the last subsections, both matrices are calculated using information from the labeled data in the training set $\mathcal{X}^{(l)}$. However, the way in which this information is extracted is different for each matrix. For the similarity matrix, \mathcal{W}_{sim} , all training data are treated together in a single network in order to extract global information of the entire set. Thus, all classes are mixed into this single network. On the other hand, the structural term, \mathcal{W}_{str} , has a different purpose: to extract and measure the pattern formation of each class. Thus, instances of different classes must be treated separately. After composing the modified weight matrix in step 4, the limiting probabilities of the random walk process are calculated (step 5) and $\mathbf{x}^{(u)}$ receives the label from the class with the largest probability (steps 6 and 7).

Algorithm 4 High-level network-based data classification

Input:

$\mathcal{X}^{(l)}$: training data set

$\mathbf{x}^{(u)}$: unlabeled instance

Parameters:

α : parameter for the convex combination in Eq. 3.3

τ : number of largest probabilities used for classification

Output:

l : estimated class label for $\mathbf{x}^{(u)}$ ($l \in \mathcal{L}$)

Training:

1 : \mathcal{W} = Create a network using $\mathcal{X}^{(l)}$

Classification:

2 : \mathcal{W}_{sim} = Compute similarity matrix (Eq. 3.5)

3 : \mathcal{W}_{str} = Compute structural matrix (Eq. 3.7)

4 : $\hat{\mathcal{W}}$ = Compose modified weight matrix (Eq. 3.3)

5 : \mathbf{p}^∞ = Compute limiting probabilities (Eq. 3.4)

6 : \mathcal{T} = Select the τ -largest probabilities from \mathbf{p}^∞

7 : l = Assign $\mathbf{x}^{(u)}$ the most representative class in \mathcal{T}

The computational complexity of the proposed technique can be analyzed in terms of steps 1 through 6 of Alg. 4 as follows. The creation of a graph in step 1 requires a computation of $O(n^2)$ since the similarity between all pair of instances must be calculated. In step 2, the weight biases composition needs $O(n^2)$ operations since each matrix entry must be added. The computation of clustering coefficient in step 3 is $O(n\langle k \rangle^2)$, where $\langle k \rangle$ is the average degree of the network. In step 4, all matrix entries must be added up, requiring $O(n^2)$ computations. The limiting probabilities in step 5 can be promptly calculated by the iterating the system of Eq. 3.4

to the stationary state in $O(n^2)$ computations. Step 6 requires $O(n \log n)$ by using an efficient sorting algorithm such as the Quicksort algorithm (Cormen *et al.*, 2003). Putting it all together, the computational complexity of the proposed technique is $O(n^2 + n^2 + n < k >^2 + n^2 + n^2 + n \log n)$. Taking the highest order term, it is $O(n^2)$.

However, the above complexity can be reduced by optimizing the algorithm to deal with sparse networks such as the k -nearest neighbor networks (all networks constructed in the simulations of this section fall in this category), in which connection matrices are sparse ($< k > \ll n$). In this case, steps 2, 4 and 5 can be reduced to $O(n < k >)$ computations. In addition, by using the graph construction method based on Lanczos bisection (Chen *et al.*, 2009a), step 1 requires $O(n^t)$, and the complexity reduces to $O(n^t + n < k > + n < k >^2 + n < k > + n < k > + n \log n)$. According to (Chen *et al.*, 2009a), $1.06 \leq t \leq 1.33$ is sufficient to achieve high quality networks. Thus, the computational complexity order of the proposed technique varies from $O(n^{1.06})$ to $O(n^{1.33})$.

The complexity comparison with the optimized implementations of the simulated techniques shown in section 3.2.3 is as follows: i) kAOG requires from $O(n^{1.06})$ to $O(n^{1.33})$ computations (Bertini *et al.*, 2011); ii) standard k-NN-based techniques requires $O(n)$ (it can be $O(n^{0.5})$ when optimized with kd tree methods (Grother *et al.*, 1997)); iii) for C4.5, tree induction requires $O(n(\log n)^2)$ (Quinlan, 1992) ($O(cn)$ in specific cases (Su and Zhang, 2006)); iv) finally, standard SVM is $O(n^3)$, and state-of-the-art implementations have empirically a training time that scales between $O(n)$ and $O(n^{2.3})$ (Platt, 1999; Tsang *et al.*, 2005).

3.2.3 Experimental results

In this subsection, we present simulation results to show some properties and the performance of the proposed classification scheme. Starting with toy problems, the behavior of the similarity matrix and of the structural matrix in Eq. 3.3 are illustrated. The influence of parameters α and τ is also investigated. Finally, we present a comparative study to some well-known classification techniques and an application on image classification.

3.2.3.1 Toy problems

A simple numerical example is shown to illustrate only the similarity term behavior. Given the similarities in Fig. 3.3, in which $S = [0.31 \ 0.36 \ 0.20 \ 0.19]$, the matrix composition of Eq. 3.5 results in:

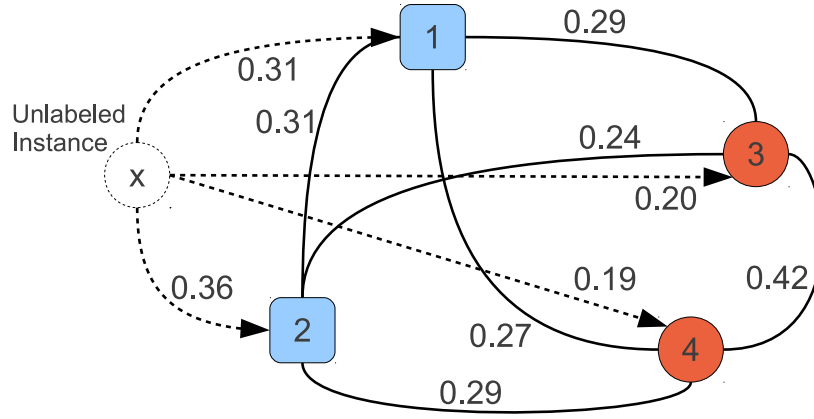


Figure 3.3: Similarity values among network nodes used in the matrix composition \mathcal{W}_{sim} of the toy example.

$$\mathcal{W}_{sim} = \begin{bmatrix} 0.31 & 0.67 & 0.49 & 0.46 \\ 0.62 & 0.36 & 0.44 & 0.48 \\ 0.60 & 0.60 & 0.20 & 0.61 \\ 0.58 & 0.65 & 0.62 & 0.19 \end{bmatrix}.$$

As we are interested in the behavior of the similarity term, we set $\alpha = 1$ in Eq. 3.3, and so $\hat{\mathcal{W}} = \mathcal{W}_{sim}$. Finally, the limiting probabilities (Eq. 3.4) are:

$$\mathbf{p}^\infty = \begin{bmatrix} 0.2665 & 0.2849 & 0.2242 & 0.2244 \end{bmatrix}.$$

We are able now to sum up the τ -largest limiting probabilities (from 1 to 4) of the unlabeled instance $\mathbf{x}^{(u)}$. The classification results are compiled in Table 3.4. The following two observations can be made for the results. First, in this simple example, the technique correctly classifies $\mathbf{x}^{(u)}$ as pertaining to blue squares class for all values of the parameter τ . The similarity biases introduced by $\mathbf{x}^{(u)}$ and propagated by the random walk process reduced the distances in the blue squares region and, as a consequence, strengthened the limiting probabilities, making them more representative in that region. Second, at the beginning, $\mathbf{x}^{(u)}$ is more similar to node 3 ($\mathcal{S}_3 = 20$) than to node 4 ($\mathcal{S}_4 = 19$), but it turned out to be slightly more similar to node 4 than to node 3 at the end of the process (with limit probability of 0.2244 against 0.2242). This behavior is due to the biased paths among nodes on the underlying network, of which global structure is taken into account by the similarity term. As can be noted in Fig. 3.3, $\mathbf{x}^{(u)}$ makes the strongest bias in the region of node 2 (0.36), which in turn has a direct link to node 4. Therefore, the random walker is influenced by the biases in the connection weight matrix. As a consequence, the transition probabilities among nodes are changed accordingly.

Table 3.4: Classification results for the toy example showed in Fig. 3.3. Results in boldface are the most representative class for each number of τ -largest probabilities selection.

τ		1	2	3	4
Corresponding nodes		{2}	{2, 1}	{2, 1, 4}	{2, 1, 4, 3}
Probability per class	Blue	0.2665	0.5514	0.5514	0.5514
	Red	0.000	0.000	0.2244	0.4486
Estimated class		Blue	Blue	Blue	Blue

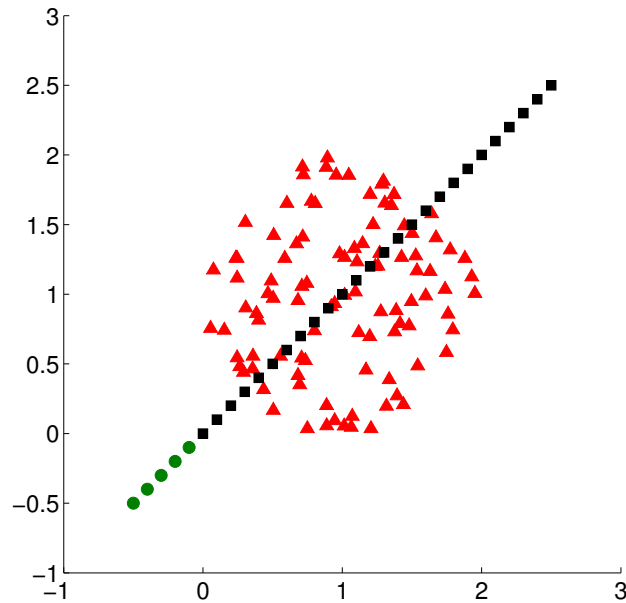


Figure 3.4: Artificial data set composed of two classes forming specific patterns. The red triangle class contains 50 training instances. The green discs class contains 5 training instances. The objective is to classify the black squares as belonging to the straight line including the green discs class.

The next toy example depicts a situation in which the structural term, \mathcal{W}_{str} , plays a key role in classification. Figure 3.4 shows a specific scenario, where two classes (green discs and red triangles) correspond to two distinct data patterns. The class represented by the green discs forms a straight line, while the class of red triangles compose a spherical shape. The objective is to classify the black squares. In this example, they complement the straight line started by the green discs, and so they should belong to that class (Fig. 3.5).

For the sake of clarity in this toy example, the classification procedure takes into account only the structural term. First, a network \mathcal{G}_i is constructed for each class separately. In the construction, a link is generated between a node and its neighbors that lie in a region limited by radius ϵ . In this example, the value $\epsilon = 0.77$ is used so that

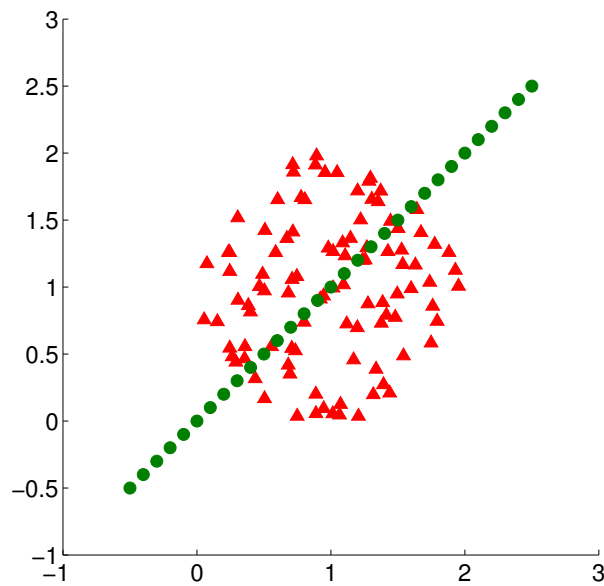


Figure 3.5: Correct classification for the data set in Fig. 3.4.

in the straight line each node can connect to its two nearest neighbors on each side (border nodes can have at most 2 connections and interior nodes, at most 4). Once the networks are constructed, the clustering coefficient (Eq. 2.3) is computed for all nodes. Second, links are created between the test instance and its neighbors that lie in the ϵ -radius, and the clustering coefficient is computed again. Finally, the test instance receives the class label i which corresponds to the label of network \mathcal{G}_i presenting the smallest variation of clustering coefficients after insertion of the unlabeled instance. In the case when a class has no instance in the ϵ -region, it is considered to have the largest coefficient variation. After being classified, the test instance is inserted into the training set of its corresponding class.

Figure 3.6 shows the total clustering coefficient variation (L^1 -norm) for each class caused by the insertion of a test instance. The black squares in Fig. 3.4 are classified one by one, from left to right, and so test instance number 1 corresponds to the leftmost square. It can be seen in Fig. 3.6 that the test instances caused abrupt changes in the structural measure on the red triangles class. The changes lied approximately in the interval 0.09 (4th instance) to 0.5 (7th instance). On the other hand, the changes in the green discs class were very small, varying approximately from 0.01 (first instance) to 0.09 (last instance). Hence, this example demonstrates that the pattern conformity of the green class can be kept after the insertion of the unlabeled instances.

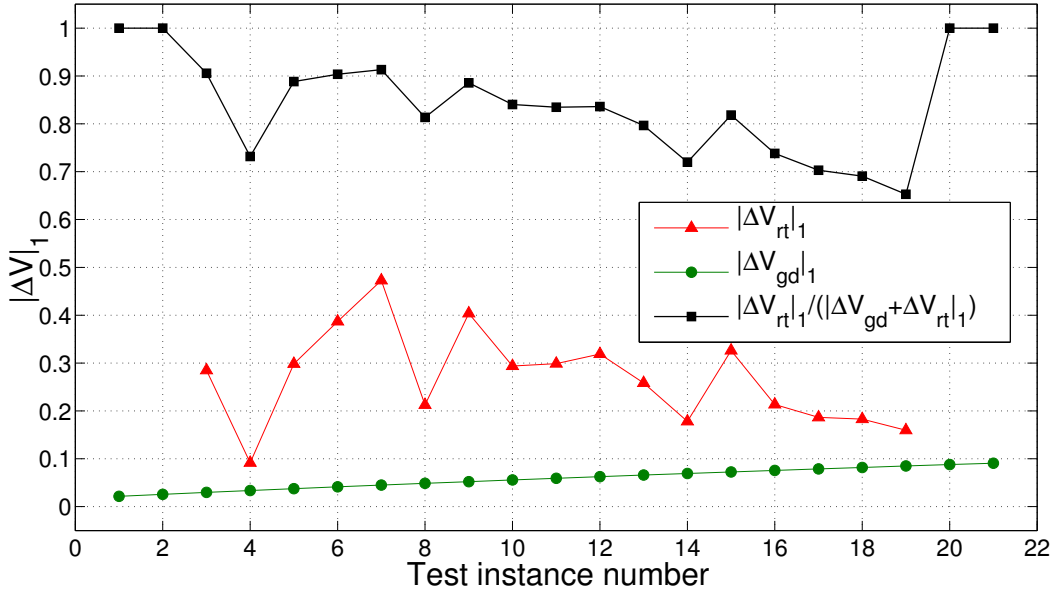


Figure 3.6: Structural changes caused by the insertion of test instances on each class in Fig. 3.4. The bottom line represents total changes ($\mathcal{L}^1 - norm$) on the green discs class ($|\Delta V_{gd}|_1$). The middle line represents changes on the red triangles class ($|\Delta V_{rt}|_1$). The top line is the relative change $|\Delta V_{rt}|_1 / (|\Delta V_{rt}|_1 + |\Delta V_{gd}|_1)$, where the two border points on each extremity accounts for the case when the test instances does not link to the red triangles class.

3.2.3.2 Effectiveness of the structural term

We assess here the effectiveness of the structural term in Eq. 3.3, that is, how this term can improve classification accuracy concerning parameter α . In the experiments, 15 real data sets were selected from the UCI machine learning repository (Bache and Lichman, 2013), and the test set of the MNIST database was used as a large data set (LeCun *et al.*, 1998). Table 3.1 (section 3.1.3) shows the metadata for all data sets. As can be seen in this table, the selection encompassed diversity on data domains as well as considered different number of classes, attributes and sizes (they vary from 3 to 15, 4 to 784 and 101 to 10,000, respectively). Eventual categorical attributes, such as in Balance and Zoo, were treated as numerical. The attribute values of all data sets were normalized to the interval $[0, 1]$. Individual cases were normalized by dividing each attribute by the square root of the sum of the squares of the individual attributes. Thus, an instance $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iq})$ was normalized by dividing each attribute x_{ij} by $\sum_{j=1}^q x_{ij}^2$.

In the simulations, parameter α was optimized in the interval $\{0.00, 0.05, 0.10, 0.15, \dots, 1.00\}$ by the grid method. Parameter τ was optimized in the interval from 1 to the number of instances of the largest class in the training data set. The networks were constructed by using the k -nearest neighbor rule, where each node links to its k -most similar neighbors, and parameter k was optimized in the

Table 3.5: Classification accuracy (%) followed by standard deviation. Each result shows the adjusted parameters for the model selection. The best results are in boldface.

Data set	Similarity term: $\alpha = 1$ (τ)	Combined terms (α, τ)	Boost (%)
Zoo	93.5 \pm 0.6 (1)	97.0 \pm 0.1 (0.90, 1)	3.7
Hayes-Hoth	59.2 \pm 1.6 (7)	61.7 \pm 2.3 (0.40, 1)	4.2
Iris	98.0 \pm 5.8 (27)	98.0 \pm 0.6 (0.95, 27)	0.0
Teaching	64.1 \pm 2.1 (1)	65.3 \pm 2.0 (0.40, 1)	1.9
Wine	82.9 \pm 1.1 (1)	87.1 \pm 1.6 (0.45, 1)	5.1
Image	74.8 \pm 0.9 (1)	75.6 \pm 0.8 (0.60, 15)	1.1
Glass	67.2 \pm 1.0 (5)	72.8 \pm 1.1 (0.75, 1)	8.3
E. Coli	84.1 \pm 0.6 (7)	85.5 \pm 0.6 (0.65, 5)	1.7
Libras	81.6 \pm 0.8 (1)	85.0 \pm 0.8 (0.05, 1)	4.2
Balance	95.5 \pm 0.5 (1)	97.2 \pm 0.6 (0.25, 24)	1.8
Vehicle	65.5 \pm 0.7 (1)	67.7 \pm 0.6 (0.00, 43)	3.4
Vowel	96.8 \pm 0.4 (1)	97.5 \pm 0.3 (0.00, 1)	0.7
Yeast	56.7 \pm 0.4 (8)	57.2 \pm 0.5 (0.60, 8)	0.9
Wine Q. Red	59.7 \pm 0.6 (1)	61.6 \pm 0.5 (0.45, 1)	3.2
Segment	92.8 \pm 0.3 (1)	93.2 \pm 0.2 (0.80, 1)	0.4
MNIST	94.8 \pm 0.1 (1)	95.1 \pm 0.1 (0.85, 4)	0.3

interval $\{2, 3, 4, \dots, 10\}$. Each experiment was performed by using a 10-fold stratified cross-validation process (Kim, 2009). In this process, the data set is split in 10 disjoint sets and, in each run, 9 sets are used as training data and 1 set is used as the test data, resulting in a total of 10 runs. The results are averaged over 30 simulations, totaling $10 \times 30 = 300$ runs.

Table 3.5 shows the best classification rates, the corresponding standard deviations and the optimal parameter values. Specifically, the second column shows results with $\alpha = 1$, that is, only the similarity weight matrix is used; the third column shows the results with $\alpha < 1$, that is, both the similarity and the structural weight matrix are used in the classification. It can be observed that the best classification accuracies for all simulated data sets are achieved when $\alpha < 1$. This means that the structural weight matrix improves the classification performance. The boost in classification accuracy is showed in the fourth column. We can see that significant large boosts, up to 8.3% for the Glass data set, were achieved.

3.2.3.3 Influence of parameter α

As have been previously showed, a mix between the similarity and the structure terms results in a better classification accuracy. So, we provide here an analysis to guide the choice of parameter α with regarding to data set features. We took 3 representative data sets from Table 3.1 to check the required α values for achieving the highest classification accuracy. Figure 3.7 shows the classification accuracy for Iris,

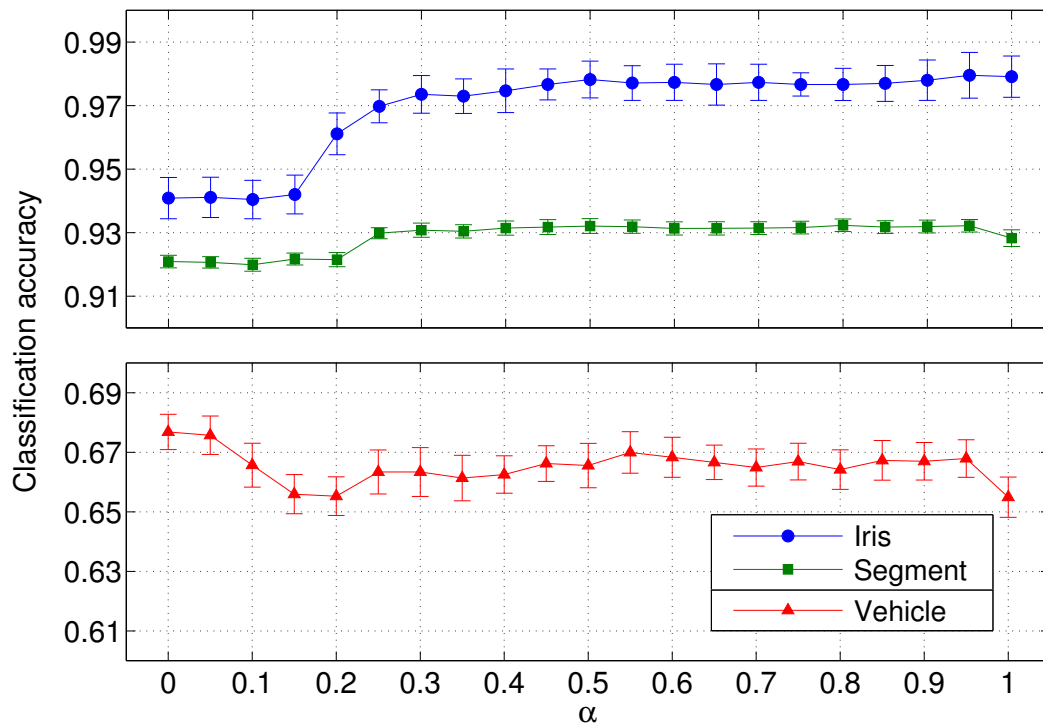


Figure 3.7: Classification accuracy in function of parameter α for Iris, Segment and Vehicle data sets.

Segment and Vehicle data sets in function of α . It can be observed that a very large value ($\alpha = 0.95$) resulted in the best accuracy for the Iris data set. In this case, the similarity term of Eq. 3.3 played the main role. For the Segment data set, a value of $\alpha = 0.80$ implied the mixing of both similarity and structural terms. Finally, for the Vehicle data set, the best accuracy required a very small value ($\alpha = 0.00$), which means that, in this case, the structural term dominated.

Now we show how data sets features influence the optimal value of parameter α . Figure 3.8 shows the class distribution for the Iris data set by using the first three largest PCA components. Clearly, the 3 classes are well defined in this case. As a consequence, the technique can rely on the similarity structure because only a small overlap occurs between any two classes. This can be achieved by setting α with a large value to give a large weight to the similarity term. As a slightly different situation, Fig. 3.9 shows the first three PCA components for the Segment data set. As can be seen, this data set is composed by 4 almost well defined classes (blue, red, light green and brown), while the other 3 classes are strongly mixed. In this case, the technique can not rely completely in the similarity structure of the data set because of the expressive overlapping cases. To avoid this situation, the technique achieved the best result by using $\alpha = 0.80$, which gives 20% of weight to the high level structural term, increasing the weight of the local measure. As the last situation, Fig. 3.10 shows the first three PCA components for the Vehicle data set. It can be observed that the 4 classes are highly overlapped with each

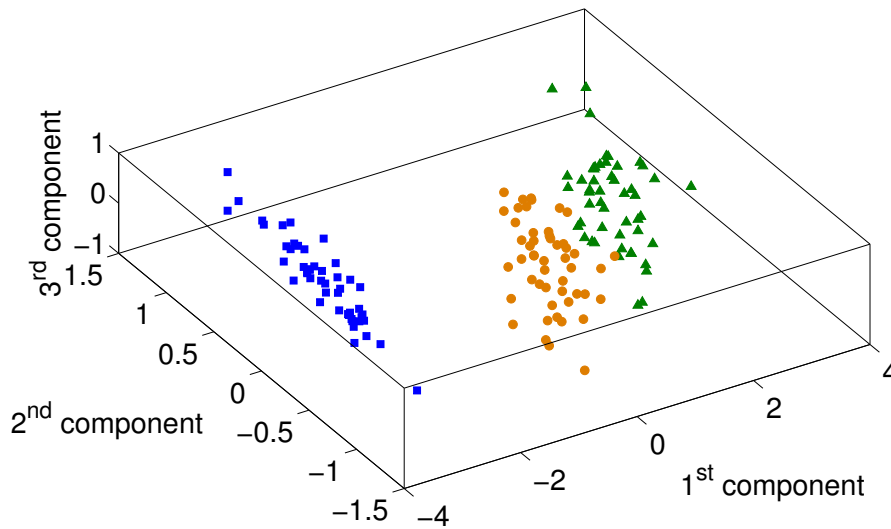


Figure 3.8: Class distribution of the Iris data set using the first three most representatives PCA components. The classes are well defined.

other. As a consequence, the technique can not rely on the similarity matrix because the links among nodes would be links among a mix of different classes. Therefore, the structural term is more reliable as the mix among classes would be decreased in this case. This is achieved by a small value for α (0.00). This study shows that the structural term is specially useful in the complex situation of classification, that is, when the low level physical feature fails to distinguish different classes.

To complete the analysis, Fig. 3.11 shows the class dispersion and the optimal α values for all data sets in Table 3.1. The data set dispersion was calculated in terms of the Fisher ratio: the ratio of the between-class scatter to the within-class scatter in the feature space. Fig. 3.11 shows the results normalized in the interval $[0, 1]$. Here, the same evidence as in the study of the three representative data sets (Iris, Segment and Vehicle) can be found. In Fig. 3.11, the Pearson's correlation coefficient between both curves is 0.52 in the interval $[-1, 1]$, being -1 a total anticorrelation and 1 a perfect linear dependency. The value 0.52 means a high correlation between the α value and the data set class dispersion.

3.2.3.4 Influence of parameter τ

In order to analyze the influence of parameter τ in the classification accuracy, we simulated three artificial data sets as depicted in Fig. 3.12. These three Gaussian data sets are composed of two classes with 500 instances each, with different levels of overlapping. In all cases, it can be seen that relatively small values ($\tau \approx 30$, representing 3% of the data set size) are sufficient to achieve the optimal or sub-optimal results. The same evidence was found for the real data sets shown in the next subsection 3.2.3.5, in

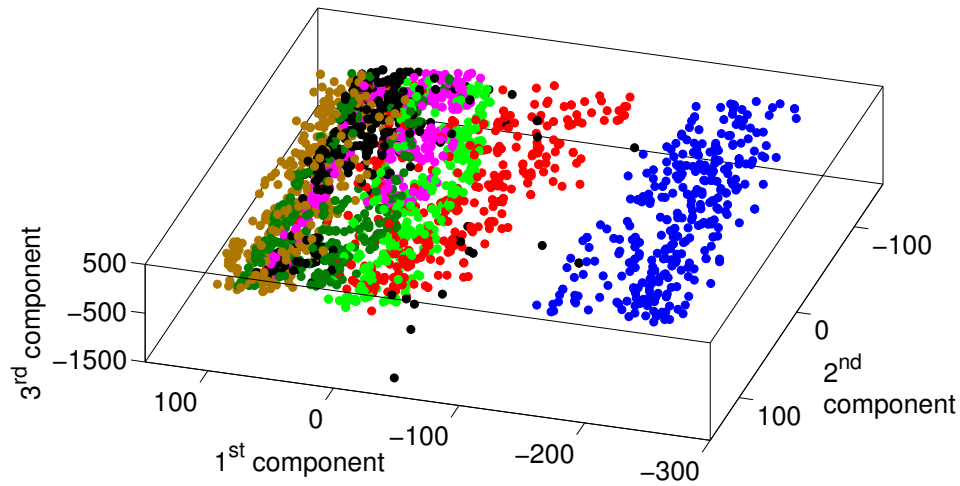


Figure 3.9: Class distribution of the Segment data set using the first three most representative PCA components. Some classes are very overlapped.

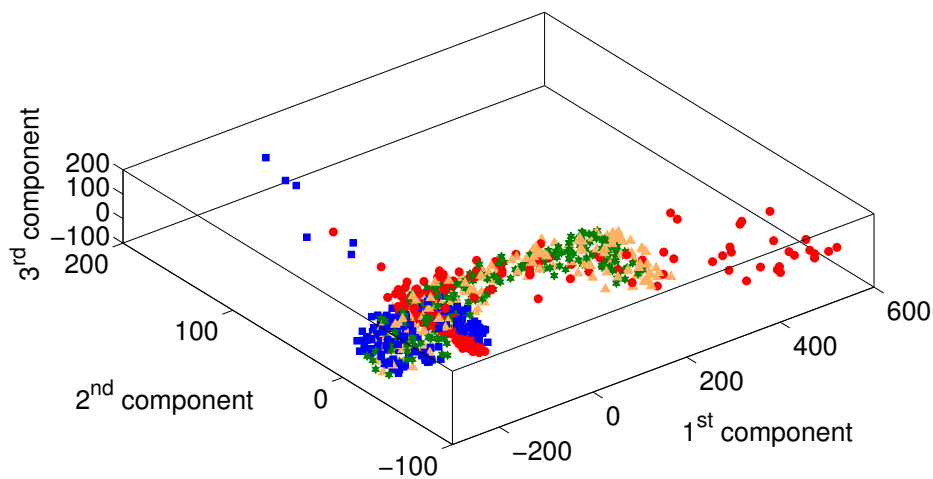


Figure 3.10: Class distribution of the Vehicle data set using the first three most representative PCA components. All classes are very overlapped.

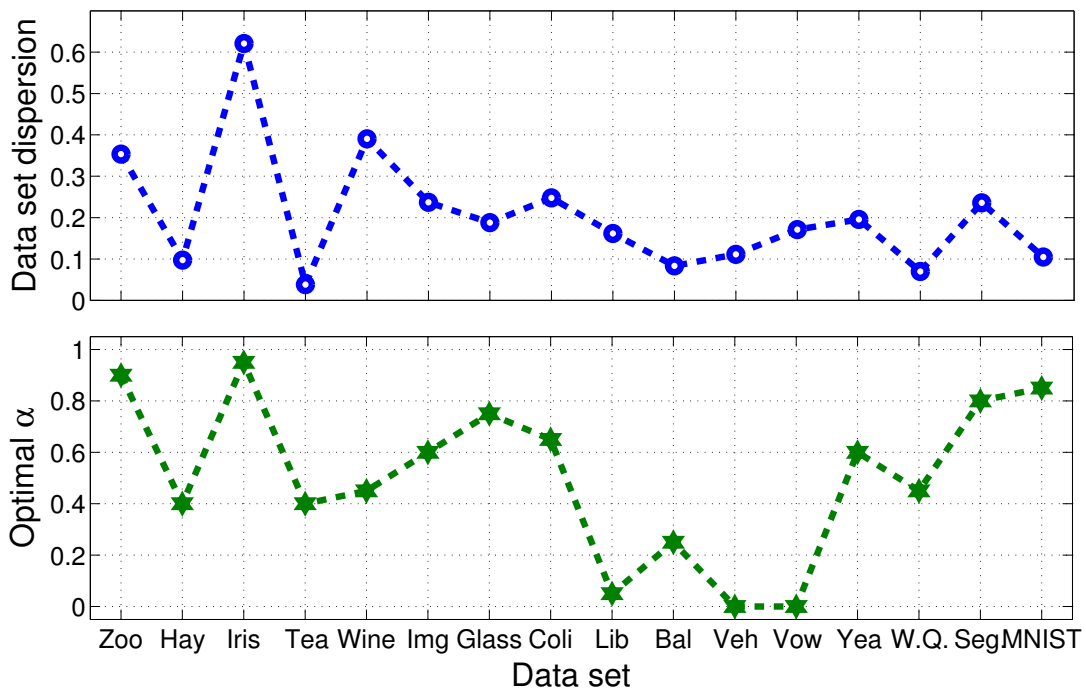


Figure 3.11: Data set class dispersion and the optimal α values for all data sets in Table 3.1. The correlation coefficient between both curves is 0.52, indicating a high linear dependency between them.

which simulation accuracies achieved the best results by using small values for parameter τ , most of them lying in the range of approximately 0.04% to 7.14% of the number of instances. Hence, a good heuristic is to choose small values for parameter τ in the range of optimization. Actually, since τ is a discrete parameter and it defines the number of neighbors used for classification, it is reasonable to assume that the parameter range of τ is very small, that is, $\tau_{max} = constant \ll n$, and so finding out the optimal value of τ by exhaustive searching does not change the complexity order of the algorithm and, in practice, it is a very quick process.

3.2.3.5 Comparative study

The proposed technique was compared to other 6 representative multi-class classification algorithms. Three of them are nearest neighbors algorithms (Tan *et al.*, 2005), namely k -Nearest Neighbors (k NN), Weighted k NN (WkNN) and Prototype-based k NN (PkNN) (Hastie *et al.*, 2009). The other three are Decision Tree C4.5 (Quinlan, 1992), Multi-Class SVM (MSVM) (Vapnik, 1999) and the network-based k -Associated Optimal Graph (k AOG) (Bertini *et al.*, 2011).

The preprocessing of data sets and the parameter adjustment for the proposed technique are described previously (subsection 3.2.3.2). For the parametric algorithms (all algorithms except k AOG), a repeated cross-validation (Kim, 2009) was performed to optimize their respective parameters. For the SVM algorithm, we used the *one-against-*

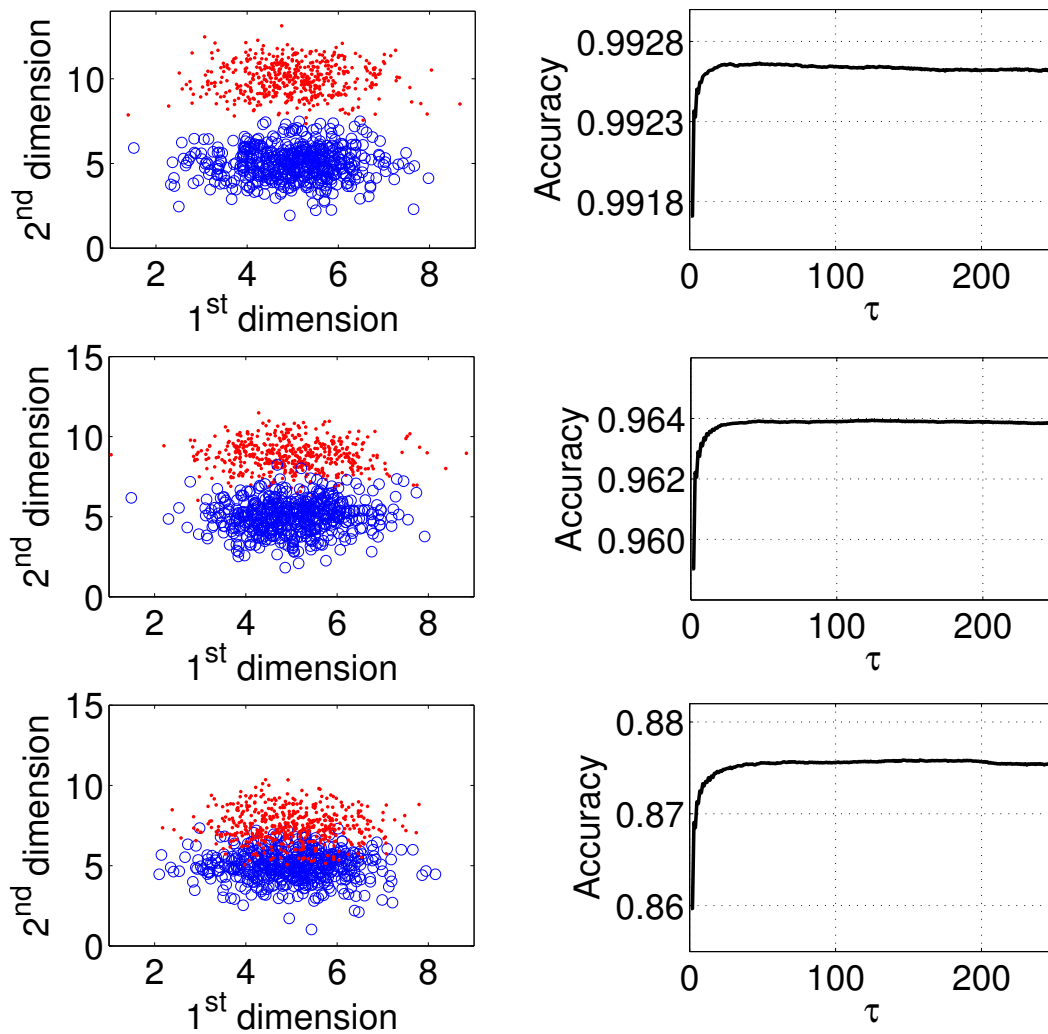


Figure 3.12: Classification accuracy in function of parameter τ . On the left column, there are three distinct artificial data sets with two classes of 500 instances each. Class distribution range from non-overlapped (first row) to highly overlapped (third row). The right column depicts the classification accuracy for each data set in function of parameter τ . It can be seen that relatively small values ($\tau \approx 30$, representing 3% of the data set size) are sufficient to achieve the best results, around 99.2%, 96.4% and 87.5% top-down.

one multi-class version, in which $\mathcal{L}(\mathcal{L} - 1)/2$ binary classifiers distinguish between every pair of classes by using a voting scheme. To avoid ties, the output of each SVM corresponds to the real valued decision functions. The *one-against-one* SVM is preferable over other multi-class methods and yields the best results among them (Hsu and Lin, 2002). To reduce the parameter search space in the SVM model selection, the kernel in consideration was only the radial basis function, $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, and the stopping criterion for the optimization method was defined as the Karush-Kuhn-Tucker violation to be lower than 10^{-3} , the same condition used in (Hsu and Lin, 2002). The model selection was performed by considering the kernel parameter $\gamma \in \{2^4, 2^3, \dots, 2^{-10}\}$ and the cost parameter $cp \in \{2^{12}, 2^{11}, \dots, 2^{-2}\}$, resulting in 125 combinations for each data set. For k NN algorithms, the only parameter is the number of neighbors k , which ranged from 1 to the number of instances in the largest class of the training set. In the Wk NN, the classification was performed by summing the weights between the instance to be labeled and its k -nearest neighbors. Specifically, the weight between two instances \mathbf{x}_i and \mathbf{x}_j was defined by $1/\|\mathbf{x}_i, \mathbf{x}_j\|$, where $\|\cdot\|$ is the Euclidean distance. For the Pk NN, the prototypes were obtained by using the k -means algorithm (Hastie *et al.*, 2009), and the number of prototypes per class was selected in the interval $[1, 30]$. For the C4.5 algorithm, two parameters were adjusted: the confidence factor, which assumed the values $cf \in \{0, 0.1, 0.25, 0.5, 0.8, 1\}$, where smaller values incurred more pruning (1 is for no pruning); and the minimum number of instances μ that a set must had to be further partitioned ($\mu \in \{0, 1, 2, 3, 4, 5, 10, 15, 20, 50\}$).

Table 3.6 presents the classification accuracies achieved by the algorithms for each data set as well as the optimized parameter values. Table 3.7 shows the corresponding rank. The ranking measure was calculated as follows: i) for each data set, a rank value is determined according to the accuracy of the technique, that is, the best technique on a given data set is ranked as 1, the second best is ranked as 2, and so on; and ii) for each algorithm, the final rank is the average rank values on all data sets. From Table 3.7, it can be seen that the proposed technique achieves the best rank among all techniques under comparison. Moreover, as it can be seen in Table 3.6, the proposed technique also exhibited the smallest deviation values.

For the sake of completeness, the statistical method proposed in (Demšar, 2006) (and references therein for the ahead mentioned tests) was used to compare multiple classifiers over multiple data sets. Following this method, the average rank of each algorithm is used in the non-parametric Friedman test to check whether they are significantly different from the average value of the overall ranks (3.93 in our simulations). The Friedman test uses the F-distribution to calculate a critical value in order to reject the null-hypothesis (all techniques are equivalent) under a defined confidence level. Being N_a the number of tested algorithms, and N_b the number of data sets, the critical value is determined by the degree of freedom $(N_a - 1, (N_a - 1)(N_b - 1))$. In our case,

Table 3.6: Classification accuracy (%) followed by standard deviation. Each result shows the adjusted parameters for the model selection. The best results are in boldface.

Data set	Proposed (α, τ)	kAOG	kNN (k)	WkNN (k)	PkNN (p)	C4.5 (cf, m)	MSVM (cp, γ)
Zoo	97.0 \pm 0.1 (0.90, 1)	97.0 \pm 5.2	96.1 \pm 5.9 (1)	96.2 \pm 5.8 (1)	93.6 \pm 7.1 (2)	95.8 \pm 5.3 (0.1, 0)	96.3 \pm 6.4 (2 ¹ , 2 ¹)
Hayes-Hoth	61.7 \pm 2.3 (0.40, 1)	55.7 \pm 12.6	54.9 \pm 12.4 (1)	56.8 \pm 13.2 (3)	44.6 \pm 11.3 (8)	46.7 \pm 10.5 (1, 0)	45.4 \pm 13.1 (2 ¹² , 2 ¹³)
Iris	98.0 \pm 0.6 (0.95, 27)	97.4 \pm 3.2	97.9 \pm 3.4 (19)	97.9 \pm 3.3 (19)	97.3 \pm 3.2 (3)	95.0 \pm 5.8 (0.25, 2)	97.0 \pm 4.6 (2 ⁻² , 2 ³)
Teaching	65.3 \pm 2.0 (0.40, 1)	62.5 \pm 11.6	59.6 \pm 10.2 (1)	63.0 \pm 12.3 (9)	58.3 \pm 9.0 (29)	58.2 \pm 14.9 (1, 0)	52.5 \pm 7.9 (2 ⁶ , 2 ³)
Wine	87.1 \pm 1.6 (0.45, 1)	83.5 \pm 8.5	84.1 \pm 8.5 (1)	84.1 \pm 8.2 (1)	81.4 \pm 11.9 (28)	91.7 \pm 6.7 (0.5, 1)	94.4 \pm 5.8 (2 ¹¹ , 2 ²)
Image	75.6 \pm 0.8 (0.60, 15)	75.3 \pm 7.5	75.3 \pm 8.2 (1)	75.4 \pm 8.2 (3)	75.5 \pm 10.2 (16)	80.7 \pm 7.6 (0.8, 3)	86.7 \pm 7.4 (2 ¹⁰ , 2 ⁻³)
Glass	72.8 \pm 1.1 (0.75, 1)	72.5 \pm 8.1	71.9 \pm 8.6 (1)	71.8 \pm 9.0 (1)	67.3 \pm 11.8 (6)	66.9 \pm 9.4 (0.1, 3)	69.5 \pm 5.6 (2 ¹⁰ , 2 ⁴)
E. Coli	85.5 \pm 0.6 (0.65, 5)	85.8 \pm 6.4	86.5 \pm 5.2 (9)	87.4 \pm 5.4 (9)	80.4 \pm 5.2 (2)	83.6 \pm 6.1 (0.25, 3)	86.7 \pm 8.2 (2 ¹² , 2 ⁻⁹)
Libras	85.0 \pm 0.8 (0.05, 1)	85.4 \pm 5.3	84.8 \pm 5.5 (1)	84.8 \pm 5.4 (1)	55.7 \pm 4.2 (13)	71.6 \pm 7.5 (0.8, 1)	86.6 \pm 5.0 (2 ⁷ , 2 ²)
Balance	97.2 \pm 0.6 (0.25, 24)	94.9 \pm 2.5	94.7 \pm 2.6 (1)	96.7 \pm 2.1 (11)	76.6 \pm 5.7 (8)	89.6 \pm 3.7 (0.5, 1)	98.2 \pm 0.9 (2 ⁷ , 2 ⁰)
Vehicle	67.7 \pm 0.6 (0.00, 43)	67.9 \pm 4.4	67.3 \pm 4.0 (3)	67.6 \pm 4.1 (5)	60.0 \pm 5.5 (10)	70.7 \pm 3.5 (0.5, 2)	84.4 \pm 3.4 (2 ¹⁰ , 2 ³)
Vowel	97.5 \pm 0.3 (0.00, 1)	98.9 \pm 0.7	97.8 \pm 1.0 (1)	98.8 \pm 0.9 (11)	96.6 \pm 1.8 (8)	78.6 \pm 4.3 (0.5, 0)	97.5 \pm 1.9 (2 ⁷ , 2 ⁰)
Yeast	57.2 \pm 0.6 (0.60, 8)	53.6 \pm 3.8	58.7 \pm 3.4 (15)	60.9 \pm 3.6 (16)	48.0 \pm 2.7 (1)	55.8 \pm 3.6 (0.1, 5)	58.9 \pm 4.8 (2 ¹¹ , 2 ⁰)
Wine Q. Red	61.6 \pm 0.5 (0.45, 1)	61.8 \pm 3.6	61.3 \pm 3.4 (1)	64.0 \pm 3.8 (19)	38.9 \pm 3.1 (27)	59.8 \pm 2.4 (1, 0)	60.4 \pm 3.2 (2 ⁹ , 2 ¹)
Segment	93.2 \pm 0.2 (0.80, 1)	93.7 \pm 1.5	93.6 \pm 1.6 (1)	93.6 \pm 1.4 (5)	60.9 \pm 3.2 (25)	95.4 \pm 1.2 (1, 0)	96.6 \pm 1.2 (2 ¹ , 2 ⁰)
MNIST	95.1 \pm 0.1 (0.85, 4)	95.1 \pm 1.7	95.3 \pm 2.3 (1)	95.7 \pm 1.1 (5)	94.3 \pm 2.2 (25)	94.5 \pm 1.2 (1, 0)	96.4 \pm 1.3 (2 ¹ , 2 ⁰)

Table 3.7: Rank of algorithms followed by standard deviation for the results in Table 3.6. The best ranks are in boldface.

Data set	Proposed (ν, τ)	kAOG	kNN (k)	WkNN (k)	PkNN (p)	C4.5 (cf, m)	MSVM (cp, γ)
Zoo	1	1	5	4	7	6	3
Hayes-Hoth	1	3	4	2	7	5	6
Iris	1	4	2	2	5	7	6
Teaching	1	3	4	2	5	6	7
Wine	3	6	4	4	7	2	1
Image	3	6	6	5	4	2	1
Glass	1	2	3	4	6	7	5
E. Coli	5	4	3	1	7	6	2
Libras	3	2	4	4	7	6	1
Balance	2	4	5	3	7	6	1
Vehicle	4	3	6	5	7	2	1
Vowel	4	1	3	2	6	7	4
Yeast	4	6	3	1	7	6	2
Wine Q. Red	3	2	4	1	7	6	5
Segment	6	3	4	4	7	2	1
MNIST	4	4	3	2	7	6	1
Average rank	2.88 ± 1.59	3.38 ± 1.63	3.94 ± 1.12	2.88 ± 1.41	6.44 ± 0.96	5.06 ± 1.91	2.94 ± 2.21

$(7 - 1, (7 - 1)(16 - 1)) = (6, 90)$. For a significance level of 5%, the critical value for our experiments is $F_{dist}(6, 90) \approx 2.20$. So, any result of the Friedman test larger than this critical value rejects the null-hypothesis. For the presented numerical results, the null-hypothesis was rejected because the Friedman test resulted in $F_F \approx 5.12$ and, therefore, the simulated techniques are not equivalent.

In the next step, the performance comparison of our technique in relation to the others is conducted by following the Bonferroni-Dunn test (Dunn, 1961). This test consists in comparing a control technique against the others. Here, the control technique is the one proposed in this paper. A critical difference (CD) is calculated for a given critical value (q_β) determined for a significance level β . Any rank difference larger than CD confirms that two techniques are significantly different. Thus, by using a significance level of $\beta = 0.05$, the critical value is $q_{0.05} = 2.64$, resulting in $CD = 2.02$. So, any ranking difference larger than 2.02 justifies the superiority of the best ranked algorithm. Using pairwise comparisons in Table 3.7, we see that our technique is better than $PkNN$ and $C4.5$ with a significance level of 5% and presents no significant difference to kNN , $WkNN$, $MSVM$ and $kAOG$. Thus, we can conclude that the proposed technique is at least comparable to the simulated techniques. It should also be noted that in the present comparison all techniques have been optimized with respect to their parameters for model selection.

3.2.3.6 Application to image classification

As a practical application, the proposed technique was applied to classify images from the ETH-80 collection (Leibe and Schiele, 2003). This data set comprises a collection of 3280 images of 8 categories: apple, car, cow, cup, dog, horse, pear and tomato (Fig. 3.13). For each category, there are 10 objects that span large in-class variations while still clearly belonging to the category. Each object is represented by 41 images from viewpoints spaced equally over the upper viewing hemisphere (at distances from 22.5° to 26.0°). For illustration, Fig. 3.14 shows the 41 images of the yellow race car class from car category.

In the simulations, the images were down-sampled from 128×128 (original size) to 32×32 to speed up processing. The image features were extracted by using the spatio-gram measure (Conaire *et al.*, 2007). Spatiograms are able to capture higher-order spatial moments. A 2^{nd} -order spatio-gram model of an object is identical to a histogram of its features, except that it also stores additional spatial information, namely the mean and covariance of the spatial position of all pixels that fall into each histogram bin. To compute similarities among images using spatio-grams, we used the discrete Bhattacharyya coefficient (Bhattacharyya, 1943; Djouadi *et al.*, 1990). Similar to the previous subsection, the results were averaged over 30 experiments, each one consisting of a 10-fold stratified cross-validation process.

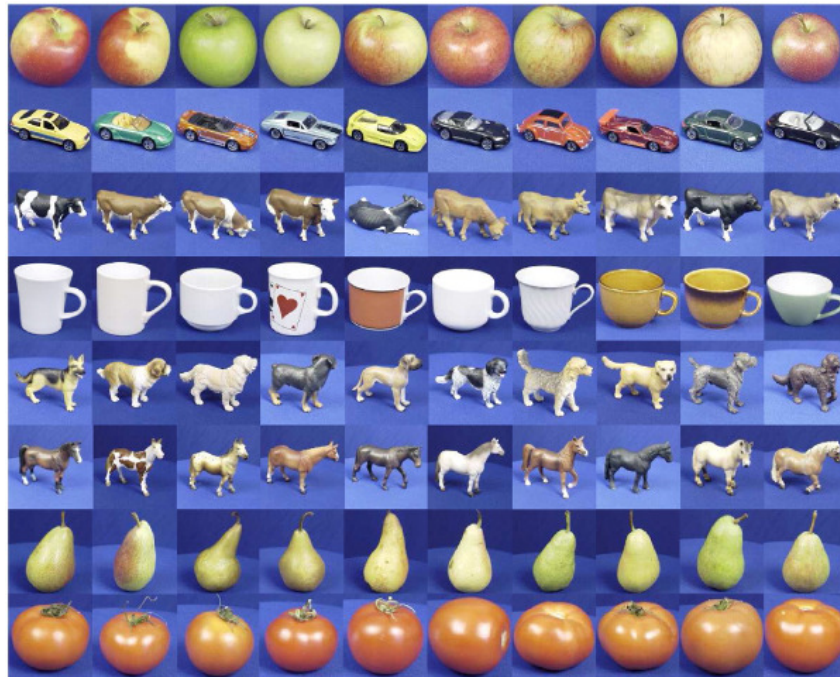


Figure 3.13: ETH-80 images collection. Each line corresponds to a category of objects. Each category comprises 10 distinct classes.



Figure 3.14: The 41 instances of the yellow race car class in the ETH-80 images collection.

Table 3.8: Classification accuracy (%) followed by standard deviation. Each result shows the parameters adjusted for the model selection. The best results are in boldface.

Category	Only similarity term: $\alpha = 1$ (τ)	Combined terms (α, τ)	Boost (%)
Apple	85.8 \pm 0.6 (1)	86.8 \pm 0.8 (0.50, 1)	1.2
Car	85.2 \pm 1.1 (1)	89.1 \pm 0.6 (0.50, 1)	4.6
Cow	61.2 \pm 1.1 (4)	65.3 \pm 1.3 (0.50, 41)	6.7
Cup	75.4 \pm 0.8 (1)	81.1 \pm 1.0 (0.40, 47)	7.6
Dog	79.3 \pm 0.8 (1)	82.1 \pm 1.0 (0.50, 39)	3.5
Horse	74.4 \pm 1.1 (1)	75.4 \pm 0.9 (0.45, 1)	1.3
Pear	69.8 \pm 1.0 (1)	74.3 \pm 0.9 (0.55, 47)	6.4
Tomato	89.9 \pm 0.9 (1)	89.9 \pm 0.8 (0.95, 1)	0.0

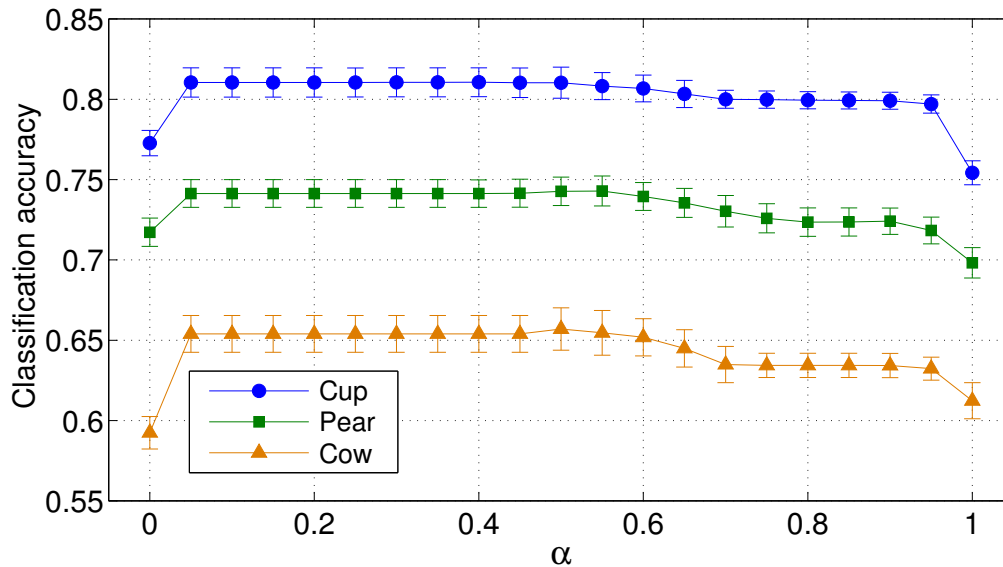






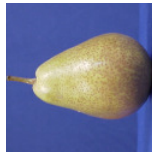




Figure 3.15: Classification accuracy of the categories Cow, Cup and Pear, in function of parameter α . As each of these categories contains very similar classes, an intermediate value for α between 0.40 and 0.60 results in the best accuracy rates.

Table 3.8 shows the simulation results. The efficacy of the technique is verified on each category separately. As each category comprises very similar classes, the classification process is a hard task. The results show that a combination of the similarity and the structural weight matrices resulted in the best classification accuracies. For illustration purpose, Table 3.9 shows some representative cases in which the usage of the structural weight matrix can correct the misclassification made by using the similarity weight matrix alone. For example, in the 2nd line of this table (Cup category), the incorrect class is very similar to the correct class of the sample image. However, when a value of $\alpha = 0.40$ was used, this error was corrected. Fig. 3.15 shows the curves of classification accuracy for the categories in Table 3.9 as a function of parameter α . We can see that, for these data sets, intermediate values of α , corresponding to a strong influence of the structural term, resulted in the best accuracies.

Table 3.9: Some representative cases of the ETH-80 classification results.

Misclassified image when $\alpha = 1$	Samples from incorrect class	Samples from correct class	α for correct classification
			0.50
			0.40
			0.55

3.3 Classification of multiple observation sets

In this section, we introduce two network-based techniques which are able to classify multiple observation sets. One of the techniques is based on the modularity measure (Cupertino *et al.*, 2012) and the other is based on the Katz centrality index (Cupertino and Zhao, 2012b). The assumption considered in both techniques is that when two groups of different pattern observations are linked together into a network, the network results in a modular structure, that is, both groups can be discriminated as subnetworks. In this situation, the modularity measure is applied directly to quantify the link strength between the subnetworks, and the Katz index measures the centrality of unlabeled observations in one subnetwork in relation to the labeled nodes in the other subnetwork.

3.3.1 The problem of multiple observation sets

The problem of multiple observation sets can be stated as follows (Kokiopoulou and Frossard, 2010). Consider a pattern \mathbf{p} has m multiple observations of the following form:

$$\mathbf{x}_i^{(u)} = o_i(\mathbf{p}), i = 1, \dots, m, \quad (3.8)$$

where o_i represents a transformation of the pattern \mathbf{p} such as rotation, scaling, perspective projection etc. Superscript u denotes that the set of observations is unlabeled, that is, the pattern the group of observations belongs to is not known. The task here is to classify \mathbf{p} into one of $|\mathcal{L}|$ classes by using its unlabeled set of observations $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)} = o_i(\mathbf{p}), i = 1, \dots, m\}$ and a group of labeled sets $\mathcal{X}_j^{(l)} = \{\mathbf{x}_i^{(l)} = o_i(\mathbf{p}_j), i = 1, \dots, n\}, j = 1, \dots, |\mathcal{L}|$, where superscript l denotes that the set of observations is labeled.

3.3.2 Classification via modularity measure

The classification technique introduced in this subsection consists of two main steps: network construction and modularity calculation (Cupertino *et al.*, 2012). The network construction is responsible for mapping the pattern observations into an underlying network structure, where links represent the similarities between two neighbor patterns, which are represented by the network vertices. The classification task is based on the modularity measure (see subsection 2.2.2). The modularity measure is capable of quantifying, based on a network scheme, the similarity between the representations of the original pattern and the set of pattern transformations. Therefore, the classification is performed by taking into account the set of transformations which is better related to the original pattern via the modularity measure.

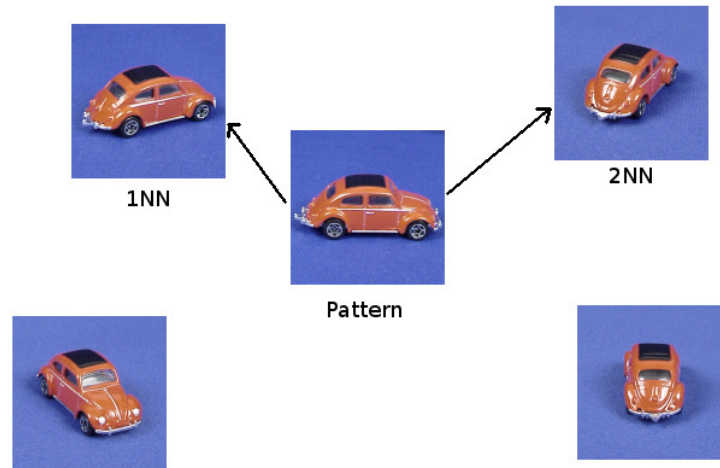


Figure 3.16: An example of a k -NN patterns network formation. The central pattern is linked with its two nearest neighbors, 1NN and 2NN.

Networks are constructed from the data sets by using the k -NN method (see section 2.2.3). This method consists in creating a link between a vertex i and its k most similar neighbors. It is equivalent to finding the k most similar patterns of a reference pattern. Consider the labeled data set $\mathcal{X}^{(l)}$ of a specific pattern \mathbf{p} . First, the similarities among all patterns must be calculated, for instance, by using the Euclidean distance. As an illustration, Fig. 3.16 shows a pattern that is linked to its two nearest neighbors. In this kind of network, if a pattern i has pattern j as its nearest neighbor, the reciprocal is not true, that is, pattern i could not be the nearest neighbor of pattern j . Therefore, in this work the edges are directed, representing only one direction of similarity. The constructed network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ does not contain self-loops. In this network, vertices represent the pattern observations in the set $\mathcal{V} = \{\mathbf{x}_i\}$, and $\mathcal{E} = \{w_{ij}\}$, $i, j = 1, \dots, n$, represent links among patterns. If there is a link between patterns \mathbf{x}_i and \mathbf{x}_j then $w_{ij} = 1$, otherwise, $w_{ij} = 0$.

Each pattern set is represented by one exclusive network. Hence, if we are given $|\mathcal{L}|$ different pattern groups, the classification technique constructs $|\mathcal{L}|$ different networks, each one corresponding to a single labeled class $\mathcal{X}_j^{(l)}$. This step can be viewed as the training phase.

The technique then computes the modularity measurement associated with the unlabeled set $\mathcal{X}^{(u)}$ concerning each one of the $|\mathcal{L}|$ labeled networks. The unlabeled set $\mathcal{X}^{(u)}$ joins into each labeled network, forming $|\mathcal{L}|$ new networks, using the same k -NN network formation method: each unlabeled pattern observation $\mathbf{x}_i^{(u)}$ is connected to its k -most similar neighbors. These similar neighbors can be selected from the set of labeled vertices $\mathcal{X}_j^{(l)}$ as well as from the unlabeled observations set $\mathcal{X}^{(u)}$, but self-connections are forbidden. Next, the modularity measure is calculated for each one of the $|\mathcal{L}|$ new networks, according to Eq. 2.5 in the subsection 2.2.2, by setting $g = 2$ densely connected subgroups, that is, one is the group of the vertices corresponding

to the initial constructed labeled network ($\mathcal{X}_j^{(l)}$) and the other group corresponds to the unlabeled vertices in $\mathcal{X}^{(u)}$ that has been joined into the labeled network. This process is performed for each one of the $|\mathcal{L}|$ labeled networks. At the end of the process, the unlabeled group is classified as belonging to the pattern network whose modularity measure has presented the lowest value, indicating that the unlabeled patterns are more similar to the patterns in that network. This process is described in Algorithm 5.

3.3.3 Classification via Katz index

The technique for classification of multiple observation sets via Katz index, introduced in Cupertino and Zhao (2012b), performs similar steps to the modularity technique previously described in the last subsection: first, the training networks are constructed by the k -NN method; second, the unlabeled observations $\mathcal{X}^{(u)}$ are joined into each labeled network, forming $|\mathcal{L}|$ new networks; finally, the Katz index is calculated for each new network according to Eq. 2.8 in section 2.2.2. At the end of the process, the unlabeled group is classified as belonging to the same class of the network of which C^{Katz} has presented the largest value, indicating that the unlabeled patterns are more similar to the patterns in that network or, in other words, that the unlabeled patterns are more central in that network. This process is described in Algorithm 6.

Given the special case of the problem at hand, just some elements of the matrix C^{Katz} are taken into account. These elements correspond to the unlabeled set of transformations $\mathcal{X}^{(u)}$. Only the centralities of the vertices that represent these elements are considered: considering two subgroups in the network, that is, the group of labeled vertices of the labeled network and the group of the unlabeled vertices joined into the labeled network, the centralities taken into account are those which compute the paths from the labeled to the unlabeled subnetworks.

3.3.4 Algorithm and complexity analysis

In a concise form, the technique for classification of multiple observation sets via modularity measure can be summarized by Algorithm 5.

The computational complexity of Alg. 5 is determined mainly by the k -NN network construction in the training phase (step 1). It takes $O(n^2)$, where n is the number of labeled observations, due to the similarity calculation among all patterns. In the classification phase, each network construction takes $O(nm)$ (step 2), where m is the number of unlabeled observations. The modularity calculation (step 3) runs in logarithmic time ($O(\log k_{max})$), where k_{max} is the maximum vertex degree in the network (Dasgupta and Desai, 2013). Therefore, the algorithm runs in $O(n^2) + O(nm) + O(\log k_{max})$. On the other hand, as the training phase must run only once before usage, the method takes $O(nm) + O(\log k_{max})$ to classify a new set of multiple observations.

Algorithm 5 Classification of multiple observation sets via modularity**Input:** $\mathcal{X}_j^{(l)}$: labeled pattern sets ($j = 1, \dots, |\mathcal{L}|$) $\mathcal{X}^{(u)}$: unlabeled observations set**Parameters:** k : number of nearest neighbors**Output:** l : estimated class for the unlabeled observations ($l \in \mathcal{L}$)**Training:****for** $j = 1 \rightarrow \mathcal{L}$ **do** 1 : $Net_j^{(l)}$ = create k-NN network from labeled set $\mathcal{X}_j^{(l)}$ **Classification:****for** $j = 1 \rightarrow |\mathcal{L}|$ **do** 2 : $Net_j^{(l+u)}$ = join unlabeled set $\mathcal{X}^{(u)}$ into $Net_j^{(l)}$ 3 : Q_j = compute modularity of $Net_j^{(l+u)}$ 4 : $l = \text{argmin}_j(Q_j)$

The technique for classification of multiple observation sets via Katz index can be summarized by Algorithm 6.

The computational complexity of Algorithm 6 is determined mainly by the k -NN network construction in the training phase and by the Katz index calculation in the classification phase. The k -NN network construction takes $O(n^2)$ (step 1), where n is the number of labeled patterns, due to the similarity calculation among all patterns. In the classification phase, each joined network runs in $O(nm)$ (step 2), where n is the number of unlabeled patterns. In the step 3, the matrix inversion required to calculate the Katz centrality takes $O((n+m)^3)$. Hence, the algorithm runs in $O(n^2) + O(nm) + O((n+m)^3)$. However, there are some methods that can reduce this cubic complexity as, for example, by taking advantage of the matrix symmetry or making approximations and decompositions to achieve an order of $O(n+E)$, where E is the total number of links in the network (Foster *et al.*, 2001).

3.3.5 Experimental results

We tested the previously introduced techniques, classification via modularity and via Katz, on two data sets. First, the results on a manuscript digits database are shown. This collection contains 20×16 binary images of handwritten digits from “0” to “9”. Each of these 10 classes contains 39 examples. Some examples can be seen in Fig. 3.17. The simulation settings was configured as follows (the same setting as in Kokiopoulou and Frossard (2010)). The data sets were split into training and test sets on each run. The training sets were composed of two examples per class. Each example was aug-

Algorithm 6 Classification of multiple observation sets via Katz centrality**Input:** $\mathcal{X}_j^{(l)}$: labeled pattern sets ($j = 1, \dots, |\mathcal{L}|$) $\mathcal{X}^{(u)}$: unlabeled observations set**Parameters:** k : number of nearest neighbors**Output:** l : estimated class for the unlabeled observations ($l \in \mathcal{L}$)**Training:****for** $j = 1 \rightarrow \mathcal{L}$ **do** 1 : $Net_j^{(l)}$ = create k -NN network from labeled set $\mathcal{X}_j^{(l)}$ **Classification:****for** $j = 1 \rightarrow |\mathcal{L}|$ **do** 2 : $Net_j^{(l+u)}$ = join unlabeled set $\mathcal{X}^{(u)}$ into $Net_j^{(l)}$ 3 : C_j^{Katz} = compute Katz centrality of $Net_j^{(l+u)}$ 4 : $l = \operatorname{argmax}_j(C_j^{Katz})$

mented by four new virtual examples generated by successive rotations of the original example. The rotation angles were sampled regularly in the interval $[-40^\circ, 40^\circ]$. The test sets were composed of m randomly chosen examples and each one was rotated by a randomly angle in the same interval, $[-40^\circ, 40^\circ]$, uniformly. These virtual examples were generated to account for the robustness to pattern transformations. The use of these examples can reinforce the transformation invariance into classification algorithms. Therefore, the classification method becomes more robust to transformations of the test instances. Values of $m = 10, 30, 50, 70, 90, 110, 130$ and 150 were considered for the test set size.

Figure 3.18 shows the classification error on the test set by varying parameter k of the k -NN network formation method. It can be seen that the higher the value for parameter k , the higher the classification error. When k takes small values, the subnetworks are well defined, meaning that the techniques identify more easily the subgroup of unlabeled observations in relation to the labeled observations group. On the other hand, when k takes large values, it is hard to discriminate the two groups. In other words, the proposed techniques must use small values for k to achieve the best results.

We compared the proposed techniques to two other network-based methods: Label Propagation (LP) (Zhou *et al.*, 2003) and Manifold-Based Smoothing under Constraints (MASC) (Kokiopoulou and Frossard, 2010). The LP algorithm forms a k -NN network with weighted edges, and computes a real-valued label matrix via a regularization framework function with a cost function (Zhou *et al.*, 2003). The idea is to find a label matrix which is smooth along the edges of similar pairs of vertices and, at the same



Figure 3.17: Examples from the handwritten digit images data set. Each class, from “0” to “9”, corresponds to a single pattern.

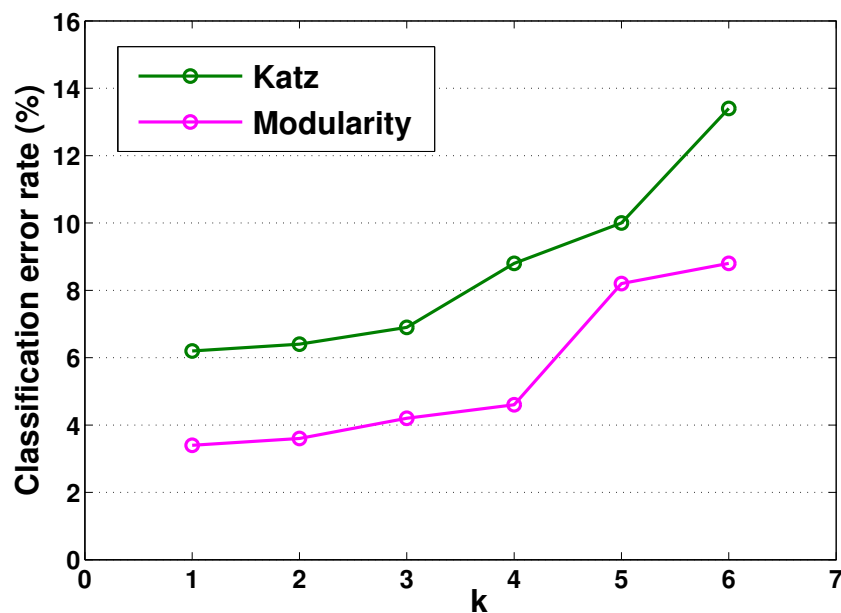


Figure 3.18: Classification error rates (%) using different values for parameter k of network construction on the handwritten digits data set. Each value was averaged over 50 runs. The number of observations for the test set is $m = 30$.

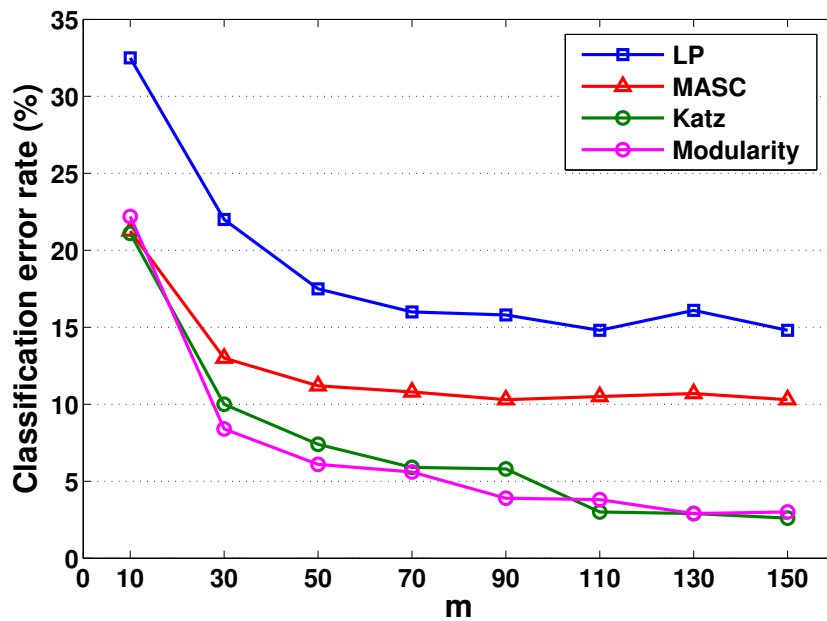


Figure 3.19: Classification error rates (%) for the handwritten digit images data set. All techniques used $k = 5$ for the network construction. Each point was averaged over 1000 runs.

time, close to the initial labels. The MASC algorithm is a specialized version of the LP algorithm to deal with the problem of multiple pattern observations sets. Since all test samples belong to the same class, the optimal solution can be obtained with a full search, as long as the number of classes stays reasonable (Kokiopoulou and Frossard, 2010). MASC has been formulated as a discrete optimization problem.

The results are shown in Fig. 3.19. For each value of m , 1000 runs were averaged, corresponding to 100 runs for each of the 10 classes. For all algorithms, we set $k = 5$, the value for which the LP algorithm achieved its best results. The proposed techniques achieved the best results for most numbers of observations m of the test sets. Furthermore, the results were even better when k took smaller values (Fig. 3.18).

The second data set used in simulations was the image collection of multiple views of objects called ETH-80 data set (Leibe and Schiele, 2003). All patterns of this set are shown in Fig. 3.13 in subsection 3.2.3.6. In total, there are 80 different classes, each one composed of 41 different views, totaling 3280 instances. Each class belongs to one of the following categories: apple, pear, tomato, cow, dog, horse, cup and car; and each category has 10 different classes. As an example, multiple observations of the class “yellow race car” can be viewed in Fig. 3.14. It must be stressed that the invariant patterns in this data set suffer from different rotation angles in three dimensions, configuring occlusions and projective distortions. See subsection 3.2.3.6 for a more detailed description about the usage of this data set.

For the ETH-80 data set, the proposed techniques were compared to four methods specialized in multiple image observations: MASC (Kokiopoulou and Frossard,

Table 3.10: Classification accuracy average (%) and standard deviation for the ETH-80 images data set.

MASC	MSM	KMSM	KLD	Katz	Modularity
88.88 ± 1.71	74.88 ± 5.02	83.25 ± 3.40	52.50 ± 3.95	92.79 ± 5.85	92.71 ± 2.65

2010) - previously described, Mutual Subspace Method (MSM) (Fukui and Yamaguchi, 2005), Kernel Mutual Subspace Method (KMSM) (Sakano and Mukawa, 2000) and KL-divergence (KLD) (Shakhnarovich *et al.*, 2002). In a few words, the last three methods work as follows:

- **MSM:** A subspace analysis method. It represents each image set by a subspace spanned by the principal components, i.e., eigenvectors of the covariance matrix. The comparison of a test image set with a training one is then achieved by computing the principal angles between the two subspaces. In the experiments, the number of principal components has been set to nine, which has been found to provide the best performance (Kokiopoulou and Frossard, 2010).
- **KMSM:** A nonlinear extension of the MSM method. Instead of using the PCA to compute the image principal components, it uses kernel PCA to take into account nonlinearities.
- **KLD:** It formulates the problem of classification of multiple observations of images as a statistical hypothesis test. Each set is assumed to fit a Gaussian distribution and the method computes the KL-divergence among the sets. The energy cut-off, which determines the number of principal components used in the regularization of the covariance matrices, has been set to 0.96. Since the KLD method relies on density estimation, it is sensitive to the number of the available data.

Table 3.10 shows the results. Both techniques, modularity and Katz, achieved the best accuracies. As in the previous experiments, $k = 5$ was used in the simulations.

3.4 kAOG embedding for dimensionality reduction

Techniques for dimensionality reduction often lie in the unsupervised or in the supervised learning. A classical example of unsupervised technique is the Principal Component Analysis (PCA) (Jolliffe, 2002). PCA is an orthogonal transformation that represent data by using the so called principal components. Usually, a small number of principal components is sufficient to account for most of the structure in the data. It maximizes the mutual information between the original high-dimensional Gaussian distributed measurements and the projected low-dimensional measurements. As an

unsupervised technique, PCA does not use the class label information of the input data. In the supervised setting, data instances are marked with label information that guides the formation of the low-dimensional space. The labels often take discrete class values, indicating which data points have to be grouped together (same class) or set far apart from the other (different classes) in the embedded space. In the group of supervised techniques, Linear Discriminant Analysis (LDA) (Fukunaga, 1991) plays an important role. As a supervised technique, it uses the class label information of the input data. LDA finds a projection matrix that maximizes the trace of the between-class scatter matrix and minimizes the trace of the within-class scatter matrix in the projected subspace simultaneously.

The network-based technique introduced in Cupertino *et al.* (2013a) comprises two complementary parts: a general graph-embedding framework for dimensionality reduction (Yan *et al.*, 2007) and the kAOG network formation method (Bertini *et al.*, 2011). Specifically, the kAOG method is extended to provide both a must-link network and a penalty network from the labeled data set $\mathcal{X}^{(l)}$ that is embedded into the general framework for dimensionality reduction.

Consider that it is given a training data set $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, containing labeled instances, and a test data set $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, containing unlabeled instances. Each instance is described by q attributes, that is, a vector $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{iq}]^T$, and belongs to a single class $l \in \mathcal{L}$, where $|\mathcal{L}|$ is the number of classes. The goal of the proposed technique is to perform dimensionality reduction by using the information provided by the labeled data set $\mathcal{X}^{(l)}$ in order to improve classification accuracy or, at least, to speed up the classification process of the unlabeled data set $\mathcal{X}^{(u)}$ without decreasing the accuracy, given that a small number q' of projected attributes is used ($q' < q$).

3.4.1 The Yan's graph-preserving criterion

Usually, the feature dimension q in real data is very high, and transforming the data from the original high-dimensional space to a low-dimensional space can alleviate the curse of dimensionality (Duda *et al.*, 2000). To accomplish that, a technique should find a mapping function F that transforms \mathbf{x} into the desired low-dimensional representation \mathbf{y} , so that $\mathbf{y} = F(\mathbf{x})$ ($\mathbf{y} \in \mathbb{R}^{q'}$). By using an underlying network to find such function F , the dimensionality reduction process can be viewed as a graph-preserving criterion of the following form (Yan *et al.*, 2007):

$$Y^* = \arg \min \sum_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij} = \arg \min Y^T L Y, \quad (3.9)$$

constrained to $Y^T B Y = \mathbf{d}$. In this formulation, \mathbf{d} is a constant vector, W_{ij} is the adjacency matrix of the network, B is the penalty matrix and L is the Laplacian matrix. The Laplacian matrix can be found via the following operation:

$$L = D - W, D_{ii} = \sum_{i \neq j} W_{ij}, \forall i.$$

The penalty matrix B can be viewed as the adjacency matrix of a penalty network W^P , so that $B = L^P = D^P - W^P$. The penalty network conveys information about which vertices should not be linked together, that is, which instances should be far apart after the dimensionality reduction process. The similarity preservation property from the graph-preserving criterion has a two-fold explanation. For larger similarity between instances \mathbf{x}_i and \mathbf{x}_j , the distance between \mathbf{y}_i and \mathbf{y}_j should be smaller to minimize the objective function. Likewise, smaller similarity between \mathbf{x}_i and \mathbf{x}_j should lead to larger distances between \mathbf{y}_i and \mathbf{y}_j for minimization (Yan *et al.*, 2007).

It is assumed that the low-dimensional attribute space can be found by using a linear projection such as $Y = X^T \mathbf{w}$, in which \mathbf{w} is the projection vector. The objective function in Eq. 3.9 becomes:

$$\begin{aligned} \mathbf{w}^* &= \arg \min \sum_{i \neq j} \|\mathbf{w}^T x_i - \mathbf{w}^T x_j\|^2 W_{ij} \\ &= \arg \min \mathbf{w}^T X L X^T \mathbf{w}, \end{aligned} \quad (3.10)$$

constrained to $\mathbf{w}^T X L X^T \mathbf{w} = d$. By using the Marginal Fisher Criterion (Yan *et al.*, 2007) and the penalty network constraint, Eq. 3.10 becomes:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \frac{\mathbf{w}^T X L X^T \mathbf{w}}{\mathbf{w}^T X L^P X^T \mathbf{w}}, \quad (3.11)$$

which can be solved by the generalized eigenvalue problem by using the equation $X L X^T \mathbf{w} = \lambda X L^P X^T \mathbf{w}$.

3.4.2 kAOG into the graph-embedding framework

In this technique, the previous Laplacian matrix L in Eq. 3.11 is calculated for the kAOG created using the labeled data set $\mathcal{X}^{(l)}$. The penalty network B , for which the Laplacian L^P is calculated to be used in Eq. 3.11, is constructed by a modification of the kAOG network formation method described in section 2.4.4. This penalty graph conveys information about which data instances should not be close together in the reduced feature space when they belong to different classes.

Algorithm 7 shows the construction of kAOG. After the initial setup, a loop starts to merge the subsequent k-Associated Graphs (kAG) by increasing k , while improving

Algorithm 7 k-Associated Optimal Graph

Require: data set $\mathcal{X}^{(l)}$
 $k \leftarrow 1$
 $G^{(op)} \leftarrow k\text{-associated graph}(k, X)$ (Algorithm 8)
repeat
 $lastAvgDegree \leftarrow D^{(k)}$
 $k \leftarrow k + 1$
 $G^{(k)} \leftarrow k\text{-associated graph}(k, X)$
 for all $C_\beta^{(k)} \subset G^{(k)}$ **do**
 if $\Phi_\beta^{(k)} \geq \Phi_\alpha^{(op)}$ **for all** $C_\alpha^{(op)} \subseteq C_\beta^{(k)}$ **then**
 $G^{(op)} \leftarrow G^{(op)} - \bigcup_{C_\alpha^{(op)} \subseteq C_\beta^{(k)}} C_\alpha^{(op)}$
 $G^{(op)} \leftarrow G^{(op)} \cup \{C_\beta^{(k)}\}$
 until $D^{(k)} - lastAvgDegree < D^{(k)} / k$ **return** $G^{(op)}$

Algorithm 8 k-Associated Graph

Require: k and a data set $\mathcal{X}^{(l)}$
 $E, B \leftarrow \emptyset$
for all $i \in V$ **do**
 if $j \in \Lambda_{i,k}$ & $c_i = c_j$ **then**
 $E \leftarrow E \cup e_{i,j}$
 else if $c_i \neq c_j$ **then**
 $B \leftarrow B \cup e_{i,j}$
 $C \leftarrow findComponents(E)$
for all $\alpha \in C$ **do**
 $\Phi_\alpha \leftarrow Eq. (2.22)$
 $G^{(k)} \leftarrow G^{(k)} \cup \{(\alpha(V', E', B'); \Phi_\alpha)\}$
return kAG $G^{(k)}$

the purity of the network encountered so far, until the optimal network measured by the purity degree is reached. Basically, the kAG algorithm links a vertex i to all its k -nearest neighbors that belong to the same class of i (the set denoted by $\Lambda_{i,k}$). More details about the algorithm are presented in Bertini *et al.* (2011).

Here, we propose a fast way to obtain the penalty matrix B for the kAOG as follows. Create a penalty link between i and j in B if j is one of the k -nearest neighbors of i and belong to a different class of i . Algorithm 8 shows how the links of the adjacency matrix E and the penalty matrix B are created for the kAG. It is worth noting that the penalty matrix is optimized by the purity measure too.

With the Laplacian of the data network L and the penalty network L^p at hand, we are able to calculate the projection vector \mathbf{w}^* by following Eq. 3.11 to find the reduced dimension by a linear projection.

Table 3.11: Metadata of the simulated data sets.

Data set	# Instances	Dimension
<i>sonar</i>	208	60
<i>libras</i>	360	90
<i>hill</i>	606	100
<i>musk1</i>	476	166
<i>CNAE</i>	1080	856

3.4.3 Experimental results

The technique’s performance was compared to other two well-known network formation methods, k-NN and ϵ -radius (subsection 2.2.3), into the general graph-embedding framework. After the dimensionality reduction step, the projected data set was classified by using the 1-nearest neighbor classification rule. In the experiments, we used 5 high-dimensional data sets comprising data from diverse and different nature from the UCI machine learning repository (Bache and Lichman, 2013): *sonar*, *libras*, *hill*, *musk1* and *CNAE*. The first data set, *sonar*, contains patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The *libras* data set contains movements from the visual language of hear-impaired people. From recorded videos, the movements were mapped in a representation with 90 features, with representing the coordinates of the movements. In the *hill* data set, each record represents 100 points on a two-dimensional graph. When plotted in order (from 1 through 100) as the y co-ordinate, the points create either a Hill or a Valley. The *musk1* data set describes a set of 92 molecules of which 47 are judged by human experts to be musks and the remaining 45 molecules are judged to be non-musks. To generate this data set, the low-energy conformations of the molecules were generated and then filtered to remove highly similar conformations, resulting in 476 conformations. The last data set, called *CNAE*, contains 1080 documents of free text business descriptions of Brazilian companies categorized into a subset of 9 categories cataloged in a table called National Classification of Economic Activities. Each document was represented as a vector, where the weight of each word is its frequency in the document. Table 3.11 shows their corresponding metadata.

For the k-NN network formation method, parameter k was optimized in the interval from 1 to the number of instances of the largest class in the training data set. For the ϵ -radius network formation method, parameter ϵ was optimized in the interval 5%, 10%, . . . , 100% of the average distance among instances in the training data set.

Table 3.12: Classification accuracy (%) using the reduced projected attribute space after dimensionality reduction performed by 3 underlying networks: kAOG, k-NN and ϵ -radius. The accuracy by using the original dimension is showed for comparison purposes. The best results are in boldface.

Data set	kAOG	k-NN	ϵ -radius	Original dimension
<i>sonar</i>	84.90 \pm 20.90	79.01 \pm 24.44	84.65 \pm 9.57	83.30 \pm 10.60
<i>libras</i>	85.06 \pm 9.26	73.93 \pm 17.94	84.89 \pm 7.24	84.89 \pm 5.76
<i>hill</i>	100.00 \pm 0.15	100.00 \pm 0.00	100.00 \pm 0.15	100.00 \pm 0.00
<i>musk1</i>	86.23 \pm 11.25	80.21 \pm 13.57	85.93 \pm 8.66	85.93 \pm 8.17
<i>CNAE</i>	87.30 \pm 9.08	83.63 \pm 10.02	86.19 \pm 5.34	86.24 \pm 5.61

Table 3.13: Projected low-dimension used for classification after dimensionality reduction for the results in Table 3.12. The percentages of the number of projected attributes compared to the original feature space are in parenthesis.

Data set	kAOG (%)	k-NN (%)	ϵ -radius (%)
<i>sonar</i>	44 (73.33)	50 (83.33)	38 (63.33)
<i>libras</i>	90 (100.00)	86 (95.56)	90 (100.00)
<i>hill</i>	89 (89.00)	24 (24.00)	89 (89.00)
<i>musk1</i>	126 (75.90)	165 (99.40)	126 (75.90)
<i>CNAE</i>	473 (55.26)	729 (85.16)	473 (55.26)

The kAOG method is non-parametric. Each experiment was performed by using a 10-fold stratified cross-validation process (Kim, 2009). In this process, the data set is split in 10 disjoint sets and, in each run, 9 sets are used as training data and 1 set is used as the test data, resulting in a total of 10 runs. The results are averaged over 30 runs, totaling $10 \times 30 = 300$ runs.

Table 3.12 shows the results of classification accuracy after dimensionality reduction by using 3 different network formation methods. In the last column of this table, it is showed the classification accuracy without any dimensionality reduction process for comparison purposes. It can be seen that the proposed technique achieved the best accuracy rates in all tested data sets by using a smaller feature space of the input data. Table 3.13 shows the number of attributes after dimensionality reduction for the results in Table 3.12. The proposed technique achieved dimensionality reductions up to 55.26% (*CNAE* data set) of the number of the original feature space. The other network formation methods also achieved good dimensionality reduction rates, but with a smaller classification accuracy (see Table 3.12).

A visual comparison of the proposed technique results to the original number of attributes can be seen in Fig. 3.20. For the sake of completeness, Fig. 3.21 shows the classification accuracy after dimensionality reduction performed by the proposed technique by using from 1 to the number of attributes of the original feature space. It can be seen that the highest accuracy is achieved by using a small number of projected

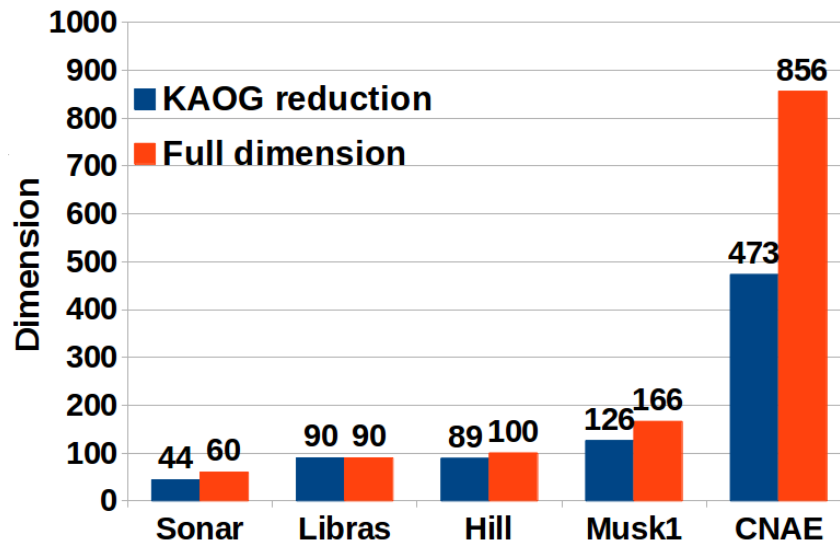


Figure 3.20: Results of the proposed dimensionality reduction technique for the 5 data sets and comparison to the original attribute space size. It can be seen that for the largest data set (CNAE) the reduction is relevant, almost half the original space.

attributes, specially for data sets *sonar* (44 attributes) and *CNAE* (473 attributes).

We also applied the proposed technique to classify a data set of images called *binary alphadigits* available online¹. This data set contains binary 20×16 digits from "0" through "9" and capital letters from "A" through "Z", with 39 sample images in each class. Figure 3.22 shows some images of this data set. In the simulations, each image was mapped as a vertex into an underlying a network.

Initially, we performed a preliminary experiment using only the images of numbers. The goal was to evaluate the potential of our technique in comparison to the other network-based algorithms. Figure 3.23 shows the results. It can be seen that the kAOG embedding dimensionality reduction technique outperformed both k-NN and ϵ -radius techniques. Also, the kAOG technique is better comparing to the classification using the original image features. For example, when using the first 150 transformed image features, out of 320, the kAOG achieved an accuracy around of 80%, against 75%, 74% and 43% when using the original features, the ϵ -radius and the k-NN respectively.

In the next experiment, the techniques were analyzed on all images (numbers + letters) from *binary alphadigits* data set, resulting totaling a size of 1014 images. Despite the higher complexity of the data set, the kAOG technique was able to perform well, according to Figure 3.24. Again, one can see that the kAOG embedding dimensionality reduction technique outperformed the other techniques, including the classification using the original image features. For example, when using the first 100 transformed image features, out of 320, the kAOG technique achieved an accuracy around of 54%,

¹<http://www.cs.nyu.edu/~roweis/data.html>

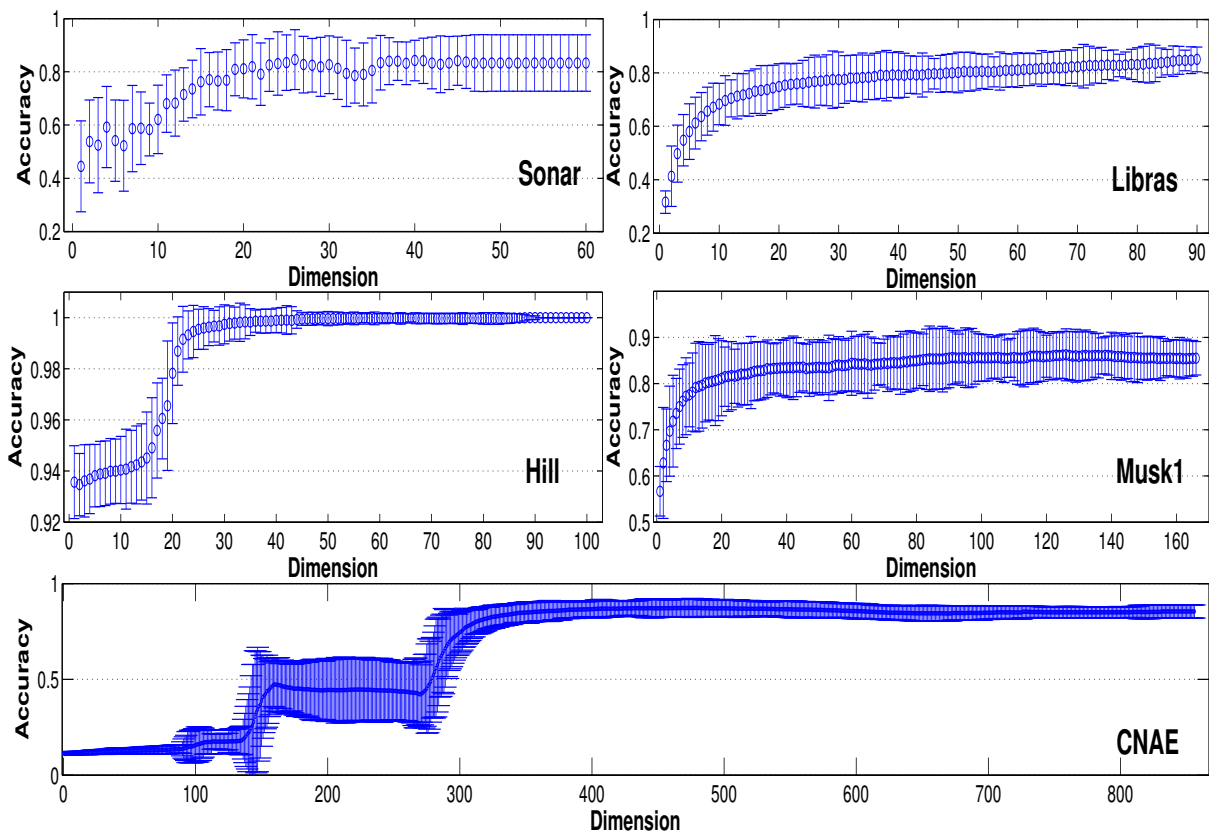


Figure 3.21: Classification accuracy by using different numbers of projected dimensions. It can be seen that the highest accuracy is achieved by using a small number of projected attributes, specially for data sets *sonar* (44 attributes) and *CNAE* (473 attributes).

against 33%, 33% and 17% when using the original features, the ϵ -radius and the k-NN respectively. These results have showed our technique based on kAOG method can be applied to dimensionality reduction problem with good results in the considered data sets.

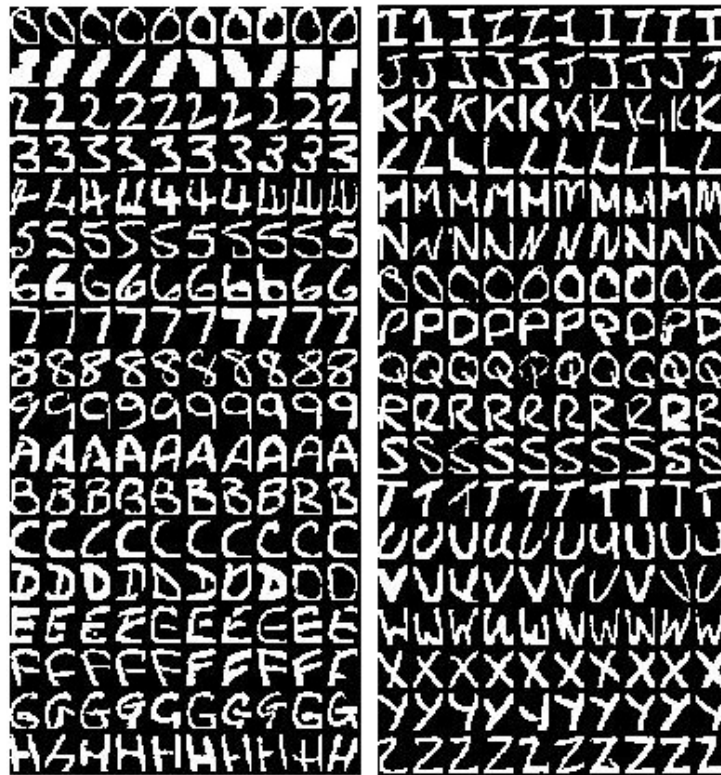


Figure 3.22: Image examples from the *binary alphadigits* data set.

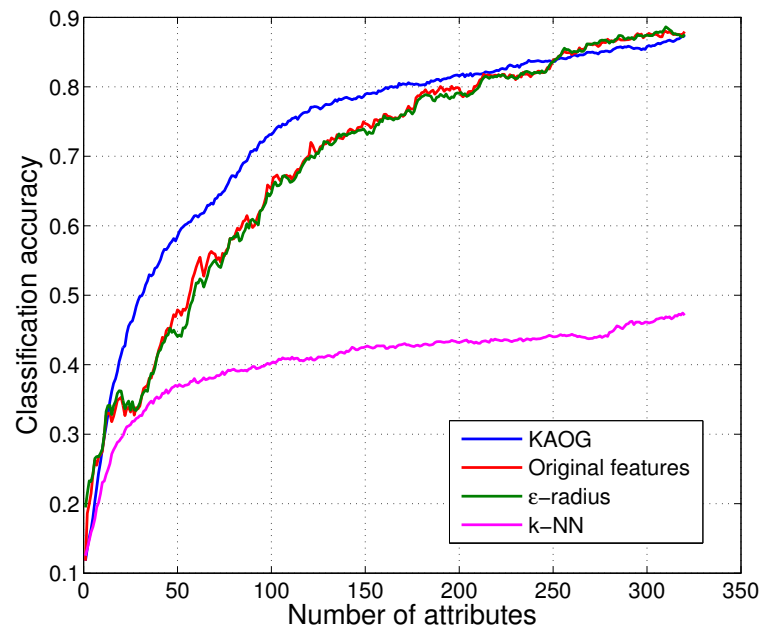


Figure 3.23: Classification accuracy on images of numbers from *binary alphadigits* data set in function of the number of transformed attributes used in classification.

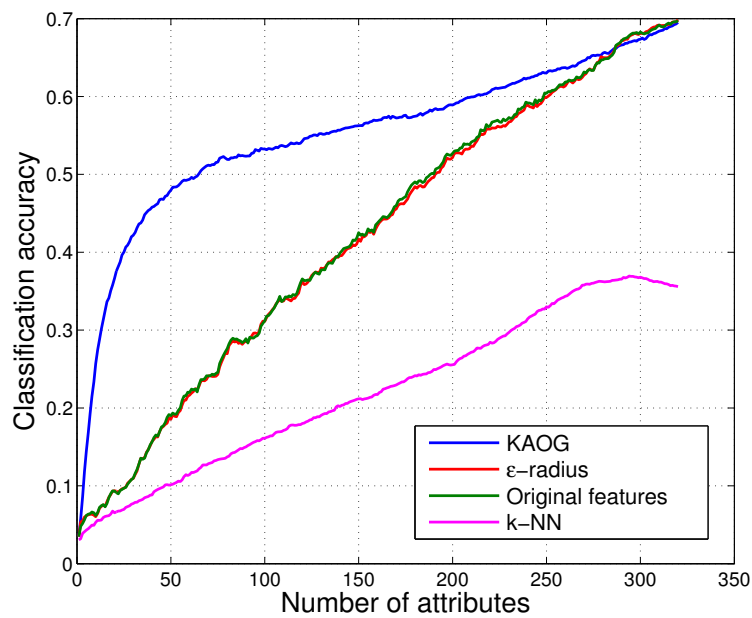


Figure 3.24: Classification accuracy on all images available into *binary alphadigits* data set in function of the number of transformed attributes used in classification. By using the kAOG network formation, the accuracy increased when using just a small number of transformed features.

Development of network-based techniques for semi-supervised learning

This chapter presents two techniques developed during the doctorate period which are related to the semi-supervised paradigm. This learning paradigm is introduced in subsection 2.1.2. As it is explained there, the problem requires a large group of unlabeled data and just a few information about labeled data. The process relies on both the massive unlabeled group and on the few labeled instances to classify the unlabeled data. In this way, the classification could achieve high accuracies even with just little knowledge about the labels.

The first technique is based on the ease of access measured by random walk limiting probabilities as it is explained in subsection 3.1, but, here, the rationale evolves on the other way round. Instead of calculating the limiting probabilities in a network of labeled vertices, the process measures the access from the unlabeled vertices to a few labeled vertices. In other words, a network is constructed from the set of unlabeled data and the few labeled data are inserted into this network to provide label biases. These biases change the network structure in a way that it influences the result of the stationary limiting probabilities: the stronger the label bias weight is over the links of a given node, the larger the limiting probability for that node is and, consequently, the greater the influence of that label is over the given node. The classification process ends by averaging the influences of all labeled vertices.

We also introduce a nature-inspired semi-supervised technique based on attraction forces. The technique models data instances as dimensionless points in a q -dimensional space and performs their motion according to the resultant force applied over them. The labeled instances act as attraction points while the unlabeled instances

receive the forces and move towards the attraction points. At a certain moment of the dynamic, the unlabeled instances receive a label that is propagated from the labeled points and become new attraction points. The model is effective and provides good classification results.

4.1 Semi-supervised transductive classification via random walk limiting probabilities

In the last subsections 3.1 and 3.2, the usage of random walk limiting probabilities to classify data was introduced. In that scheme, each node (labeled instance) is a possible state for the random walker, and the network of labeled nodes is modified by a specific link weight composition which takes into account the bias information of an unlabeled instance. The bias information changes the network structure by affecting the link weights among nodes resulting in a structure such that, after the calculation of the limiting probabilities, the most easily reached labeled nodes represent the class label of the unlabeled instance.

In this section, the above supervised classification scheme is extended to the semi-supervised paradigm. In the technique introduced in (Cupertino and Zhao, 2013b), instead of constructing a network from the labeled instances, the network uses the set of unlabeled instances to compose its nodes, and the link weight composition takes into account the information provided by the labeled instances. Here, it can be said that the random walk process measures the label propagation from labeled vertices to the remainder unlabeled vertices of the network. At the convergence, the propagation is measured by the limiting probabilities. As a smooth labeling process, each unlabeled node receives the most representative label after averaging the convergence measure.

4.1.1 Technique description

Given a data set composed of unlabeled data, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, the objective is to classify all these data samples by using a data set composed of just a few labeled instances $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, $l \in \mathcal{L}$, where $\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$, and each instance is described by q attributes $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$. Moreover, to characterize a semi-supervised learning task, $n \ll m$.

In the first step, an undirected network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ without self-loops is created by using the unlabeled data. In this network, instances are represented by vertices, $\mathcal{V} = \mathcal{X}^{(u)}$, and similarities among instances are represented by link weights, $\mathcal{E} = [\mathcal{W}_{ij}]$, $i, j = 1, \dots, m$. The network connection matrix $\mathcal{W} = \{w_{ij}\}$ is calculated by using some sort of distance function as, for example, the Euclidean distance. The entries w_{ij}

represents the similarity between a pair of instances $\mathbf{x}_i^{(u)}$ and $\mathbf{x}_j^{(u)}$, and $w_{ij} = 0$ when there is no link between $\mathbf{x}_i^{(u)}$ and $\mathbf{x}_j^{(u)}$.

In the following step, the similarity biases of the labeled instance $\mathbf{x}_j^{(l)}$ are used to change the structure of the network \mathcal{N} . To accomplish that, the similarities $\mathcal{S}_j^{(l)} = [s_{j1}, s_{j2}, \dots, s_{jm}]$ between $\mathbf{x}_j^{(l)}$ and all other vertices $\mathbf{x}_i^{(u)} \in \mathcal{V}$ are calculated by using a distance function, and a new $m \times m$ asymmetric connection matrix $\hat{\mathcal{W}}_j^{(l)}$ is composed from the initial connection matrix \mathcal{W} and the matrix of similarity biases $\hat{\mathcal{S}}_j^{(l)}$:

$$\hat{\mathcal{W}}_j^{(l)} = \mathcal{W} + \epsilon \hat{\mathcal{S}}_j^{(l)}, \quad (4.1)$$

where ϵ is a non-negative parameter and $\hat{\mathcal{S}}_j$ is the following $m \times m$ matrix:

$$\hat{\mathcal{S}}_j^{(l)} = \begin{bmatrix} \mathcal{S}_j^{(l)(1)} \\ \mathcal{S}_j^{(l)(2)} \\ \vdots \\ \mathcal{S}_j^{(l)(m)} \end{bmatrix}.$$

Remark 3. In Eq. 4.1, it can be observed that the weight biases of the labeled instance $\mathbf{x}_j^{(l)}$, encoded in matrix $\hat{\mathcal{S}}_j^{(l)}$, are applied over all edges w_{ij} of network \mathcal{N} , that is, the weight of each edge is linearly added up with the corresponding weight bias. The idea behind this operation is that the distance between any pair of vertices is reduced because of the new route introduced by the insertion of the labeled data instance. The higher the proximity between the labeled instance and a vertex, say vertex i , the more strengthened the connections from all other vertices to vertex i are. The parameter ϵ controls the influence of weight bias provided by matrix $\hat{\mathcal{S}}_j^{(l)}$ on the original network. The larger is the value of parameter ϵ , the greater will be the influence of the bias weights provided by $\mathbf{x}_j^{(l)}$.

After the weight biases composition, $\mathbf{x}_j^{(l)}$ is effectively inserted into network \mathcal{N} . An $(m+1) \times (m+1)$ weighted adjacency matrix $\mathcal{A}_j = \{a_{ij}\}$ is constructed:

$$\mathcal{A}_j^{(l)} = \begin{bmatrix} \hat{\mathcal{W}}_j^{(l)} & \mathcal{S}_j^{(l)} \\ \mathcal{S}_j^{(l)T} & 0 \end{bmatrix}. \quad (4.2)$$

In this formulation, $\mathbf{x}_j^{(l)}$ is inserted as the last entry $(m+1)$ of matrix $\mathcal{A}_j^{(l)}$, without loss of generality.

The two steps described above can be easily understood by using the toy example depicted in Fig. 4.1. In this example, a network is formed by 4 unlabeled instances

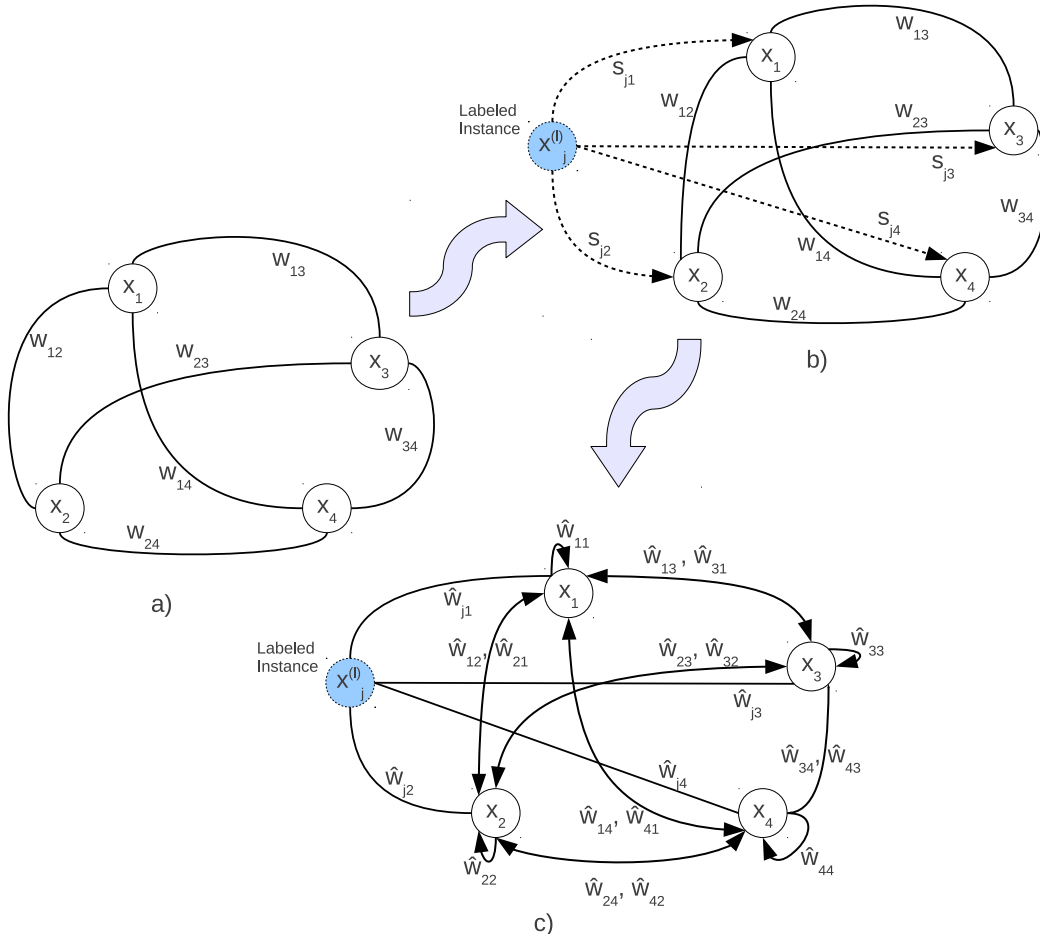


Figure 4.1: Composition process of the modified connection matrix $\hat{\mathcal{W}}_j^{(l)}$. a) An undirected and complete network \mathcal{N} is formed by using 4 unlabeled instances; b) the similarities $S_j^{(l)}$ are calculated for the labeled instance $\mathbf{x}_j^{(l)}$; c) modified network $\mathcal{A}_j^{(l)}$, directed with self-loops, after bias composition (Eq. 4.1) and insertion of the labeled vertex $\mathbf{x}_j^{(l)}$ (Eq. 4.2).

(Fig. 4.1a). In this initial network, the links are undirected and the connection matrix is symmetric. Next, the similarity vector $S_j^{(l)}$ between $\mathbf{x}_j^{(l)}$ and all other vertices is computed (Fig. 4.1b). After this computation, the weight biases of $\mathbf{x}_j^{(l)}$ are added up to the original connection matrix to form a biased connection matrix for the same network (Eq. 4.1), and $\mathbf{x}_j^{(l)}$ is effectively inserted into the network (Fig. 4.1c). It can be seen from Fig. 4.1c that the network becomes directed and with self-loops (except for $\mathbf{x}_j^{(l)}$, that has no self-loops) and the biased connection matrix is asymmetric.

After the insertion of $\mathbf{x}_j^{(l)}$ into network \mathcal{N} , we are able to compute the entries of the transition Markov matrix $\mathcal{P}_j^{(l)}$ by normalizing the rows of $\mathcal{A}_j^{(l)}$:

$$p_{jik} = a_{jik} / \sum_{k=1}^{m+1} a_{jik}, \quad i, k = 1, 2, \dots, m+1.$$

With the matrix $\mathcal{P}_j^{(l)}$ at hand, the random walk limiting probabilities can be calculated. This procedure can be performed by two ways: by finding the eigenvector corresponding to the unit eigenvalue of matrix $\mathcal{P}_j^{(l)}$, or by iterating the system

$$\mathbf{p}_j^{(l)}(t+1) = \mathcal{P}_j^{(l)} \mathbf{p}_j^{(l)}(t), \quad (4.3)$$

to the stationary state, where $\mathbf{p}_j^{(l)}$ is an $(m+1) \times (m+1)$ normalized vector. It results in the following vector:

$$\mathbf{p}_j^{\infty(l)} = [p_1 \ p_2 \ \dots \ p_{m+1}],$$

where each entry p_i represents the probability $\mathbf{x}_i^{(u)}$ belongs to the class l of the labeled vertex $\mathbf{x}_j^{(l)}$. The probability p_{m+1} is ignored as it represents the labeled vertex $\mathbf{x}_j^{(l)}$.

The above steps (Eq. 4.1 through 4.3) are repeated for each labeled instance $\mathbf{x}_j^{(l)} \in \mathcal{X}^{(l)}$, and each label probability is averaged:

$$\mathbf{p}^{\infty(l)} = \sum_{j/l=k} \mathbf{p}_j^{\infty(l)}, \quad k = 1, 2, \dots, \mathcal{L}. \quad (4.4)$$

Finally, the classification of all unlabeled instances in $\mathcal{X}^{(u)}$ is accomplished by assigning the most representative label to each unlabeled instance $\mathbf{x}_i^{(u)}$: $label(\mathbf{x}_i^{(u)}) = \underset{l}{\operatorname{argmax}} p_i^{\infty(l)}$.

4.1.2 Algorithm and complexity analysis

In a concise form, the proposed semi-supervised transductive technique can be summarized by Alg. 9.

The computational complexity of the proposed semi-supervised technique can be analyzed in terms of the steps 1 through 5 of the Alg. 9. The creation of a network in step 1 requires a computation of $O(m^2)$ since the similarities between all pair of instances must be calculated. The weight biases composition needs $O(m^2)$ operations since each matrix entry must be added in step 2. Step 3 requires a linear element insertion $O(1)$. The limiting probabilities in step 4 can be promptly calculated by the iterating the system of Eq. 4.4 to the stationary state in $O(m^2)$ computations. Step 5 of Alg. 9 is $O(n)$ in the worst case, when there is just a single available label. Putting it all together, the computational complexity of the proposed technique is $O(m^2 + m^2 + m^2 + n)$. Taking the highest order term and having in mind that $n \ll m$, it results in $O(m^2)$.

However, this complexity can be reduced by dealing with sparse networks such as

Algorithm 9 Semi-supervised transductive classification via random walk limiting probabilities

Input: $\mathcal{X}^{(u)}$: unlabeled data set $\mathcal{X}^{(l)}$: labeled data set**Parameters:** ϵ : bias weighting parameter**Output:** l : estimated class label for $\mathbf{x}_i^{(u)}$ ($l \in \mathcal{L}$)**Training:**1 : \mathcal{N} = create a network from $\mathcal{X}^{(u)}$ **Classification:****for** $\mathbf{x}_j^{(l)} \in \mathcal{X}^{(l)}$ **do**2 : $\mathcal{W}_j^{(l)}$ = compose weight biases into \mathcal{N} (Eq. 4.1)3 : $\mathcal{A}_j^{(l)}$ = insert $\mathbf{x}_j^{(l)}$ into \mathcal{N} (Eq. 4.2)4 : $\mathbf{p}_j^{(l)}$ = compute limiting probabilities (Eq. 4.3)5 : $\mathbf{p}^{\infty(l)}$ = average $\mathbf{p}_j^{\infty(l)}$ (Eq. 4.4)6 : Assign $\mathbf{x}_i^{(u)}$ the most representative label in $\mathbf{p}^{\infty(l)}$ ($\text{argmax}_l p_i^{\infty(l)}$)

the k -nearest neighbor networks, in which connection matrices are sparse (all networks constructed in the simulations of this section fall into this category). In this case, the complexity of steps 2 and 4 are reduced to $O(m \langle k \rangle)$, being $\langle k \rangle$ ($\langle k \rangle \ll m$) the average degree of the network (average number of links). Moreover, by using the graph construction method based on Lanczos bisection (Chen *et al.*, 2009b), step 1 requires $O(m^t)$, and the technique complexity reduces to $O(m^t + m \langle k \rangle + m \langle k \rangle + n)$. Furthermore, according to (Chen *et al.*, 2009b), a small value for parameter t ($1.06 \leq t \leq 1.33$) is sufficient to achieve high quality networks. Thus, the computational complexity order of the proposed technique lies in $O(m^{1.06})$ to $O(m^{1.33})$.

4.1.3 Experimental results

In this subsection, we illustrate the efficacy of the proposed technique by presenting two simulation groups. The first group comprises a toy example in which the classification task is hampered due to a mix of different shaped and mixed classes. The second group encloses some benchmark data sets used in literature for comparison purposes.

4.1.3.1 Toy example

In this subsection, we present simulation results using a toy example. A toy data set (Fig. 4.2) that captures different class characteristics, such as different shapes and

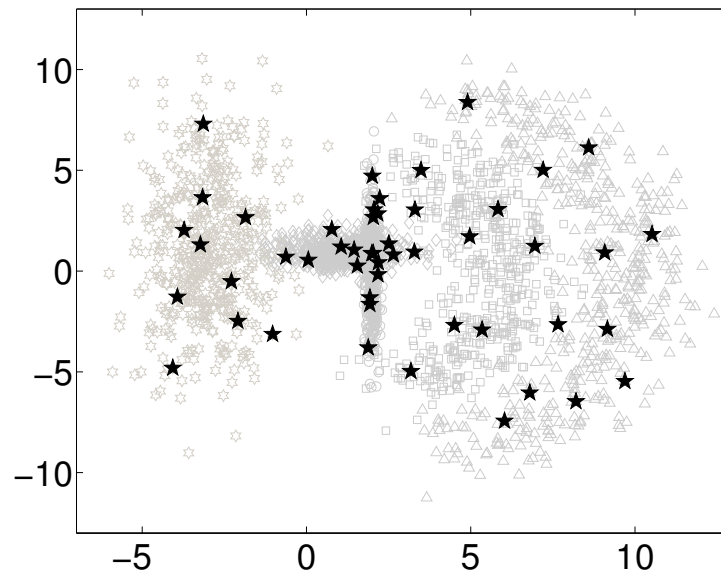


Figure 4.2: Mix of different cluster shapes from artificial data for semi-supervised classification. This artificial data set is composed of 2500 instances divided into 5 balanced and distinct clusters shapes: Gaussian, Highleyman and Lithuanian. Each cluster contains 10 labeled instances. Black-filled stars represent labeled instances.

densities, was used to encompass a challenging classification task and to illustrate the behavior of the semi-supervised technique. This data set is composed of 3 different class distributions (from left to right): Gaussian, Highleyman and Lithuanian. The data was generated by using the PRTools toolbox (Duin, 2000). Each class has 500 instances, totaling 2500 instances for the entire data set. In addition, each class comprises 10 labeled instances, representing 2% of all data. Figure 4.3 shows that the proposed technique satisfactorily labeled the 5 different classes.

4.1.3.2 Benchmark data sets

The proposed semi-supervised technique was tested and compared using 7 benchmark data sets. Table 4.1 shows a brief description of them. Three artificial sets (*g241c*, *g241d* and *Digit1*) encompass some of the semi-supervised assumptions: manifold, smoothness and cluster. The other four data sets (*USPS*, *COIL*, *BCI* and *Text*) are derived from real data. The benchmarks were developed to evaluate the power of different algorithms as neutral as possible (Chapelle *et al.*, 2006). For each data set, 24 independent splits of labeled data for the training set are available. 12 splits contain 10 labeled instances for each data set and the other 12 splits contain 100 labeled instances. For each split, at least 1 instance of each class is labeled. A more detailed explanation of each data set can be found in (Chapelle *et al.*, 2006).

The proposed technique was compared to 16 well-known and established semi-

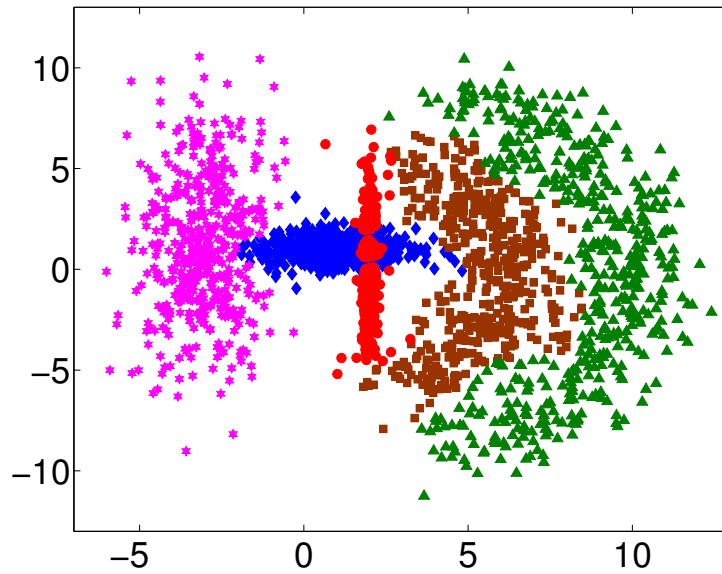


Figure 4.3: Classification achieved by the proposed technique for the data set depicted in Fig. 4.2.

Table 4.1: Meta-data of the data sets composing the SSL benchmark.

Data Set	Classes	Dimension	Points	Type
<i>g241c</i>	2	241	1500	artificial
<i>g241d</i>	2	241	1500	artificial
<i>Digit1</i>	2	241	1500	artificial
<i>USPS</i>	2	241	1500	unbalanced
<i>COIL</i>	6	241	1500	
<i>BCI</i>	2	117	400	
<i>Text</i>	2	11960	1500	sparse discrete

supervised techniques. Table 4.2 shows a brief description of them and their related references. All simulation results were extracted from (Chapelle *et al.*, 2006), where it can be found values for parameter optimization and model selection in order to minimize test errors. For LGC, LP and LNP, σ was selected from the set $\{0, 1, \dots, 100\}$ and α was fixed to $\alpha = 0.99$ (the same setup done in (Zhou and Schölkopf, 2004) and (Wang and Zhang, 2008)). For the LNP, k was evaluated for the values in $\{1, 2, \dots, 100\}$. The configuration and parameter optimization for the proposed technique were done as follows. For the network construction (Step 1 of Alg. 9), the k -nearest neighbor technique was used: each vertex was linked with its k most similar neighbors. Parameter k was evaluated for the values in $\{1, 2, \dots, 100\}$ and parameter ϵ was evaluated for the values in $\{0, 0.1, 0.2, \dots, 10\}$. In all simulations, no data preprocessing was performed by the techniques and the Euclidean distance was used to measure the similarities among data.

Tables 4.3 and 4.4 show the simulations results for 10 and 100 labeled instances, respectively, and the average rank for each technique. The ranking measure was calculated as follows: i) for each data set, a rank value is determined according to the accuracy of the technique, that is, the best technique on a given data set is ranked as 1, the second best is ranked as 2, and so on; and ii) for each algorithm, the final rank is the average rank values on all data sets. It can be seen that, for 10 labeled instances, the proposed technique achieved an average rank of 5.86 (2nd place) and, for 100 labeled instances, an average rank of 9.29 (11th place). Overall, it achieved an average rank of 7.57 (5th place) - preceded by LP (7.21), LDS (6.93), Laplacian RLS (5.43) and SGT (5.33). Interestingly, the proposed technique achieved a very high position (2nd) in the case of only 10 labeled instances, a very challenging semi-supervised task in which as only as a small portion of 0.67% of the data set is labeled. Hence, concerning the 17 techniques and the 7 benchmark data sets, the proposed technique is at least comparable to the best known semi-supervised techniques.

4.2 Semi-supervised classification based on interacting forces

The technique based on interacting forces introduced in (Cupertino and Zhao, 2012a; Cupertino *et al.*, 2013b) can be viewed as a complete network where attraction

Table 4.2: References for the simulated semi-supervised techniques.

Abbreviation	Technique	Ref.
MVU + 1-NN	Maximum Variance Unfolding	(Sun <i>et al.</i> , 2006)
LEM + 1-NN	Laplacian Eigenmaps	(Belkin and Niyogi, 2003)
QC + CMR	Quadratic Crit. and Class Mass Reg.	(Delalleau <i>et al.</i> , 2005)
Discrete Reg.	Discrete Regularization	(Zhou and Schölkopf, 2006)
TSVM	Transductive Support Vector Machines	(Chapelle and Zien, 2005)
SGT	Spectral Graph Transducer	(Joachims, 2003)
Cluster-Kernel	Cluster Kernels	(Chapelle <i>et al.</i> , 2003)
Data-Dep. Reg.	Data-Dependent Regularization	(Corduneanu and Jaakkola, 2006)
LDS	Low-Density Separation	(Chapelle and Zien, 2005)
Laplacian RLS	Laplacian Regularized Least Squares	(Sindhwani <i>et al.</i> , 2005)
CHM (normed)	Conditional Harmonic Mixing	(Burgess and Platt, 2006)
LGC	Local and Global Consistency	(Zhou and Schölkopf, 2004)
LP	Label Propagation	(Zhu and Ghahramani, 2002)
LNP	Linear Neighborhood Propagation	(Wang and Zhang, 2008)

forces represent link strength among vertices. It uses the initial labeled instances information to propagate their labels to the attracted vertices which, in turn, after being labeled, propagates their label to the subsequent attracted vertices and so on. In other words, this dynamic makes use of unlabeled data, the attracted neighbor vertices, to perform the classification task which is, in turn, the main idea of a semi-supervised technique. Even more, the model allows fine adjustment as one can use any attraction force function and adjust its parameters.

Despite the simplicity of the model, two considerations are necessary to accomplish the above mentioned behavior. One of them is to guarantee that the process is stable, and the other is to certify that the labels propagate adequately through the unlabeled instances, allowing the algorithm to converge and achieve good classification accuracy. The stability issue can be treated using similar approaches from swarm aggregation works (Gazi and Passino, 2003; Liu *et al.*, 2003), while the label propagation dynamics can be analyzed in terms of the attraction force function parameters. Both considerations are explained in the next subsections.

4.2.1 Technique description

Given a data set composed of unlabeled data, $\mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)}, i = 1, \dots, m\}$, the objective is to classify all these data samples by using a data set composed of just a few labeled instances $\mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)}, i = 1, \dots, n\}$, $l \in \mathcal{L}$, where $\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$, and each instance is described by q attributes $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}$. Moreover, to characterize a semi-supervised learning task, $n \ll m$.

The instances are modeled as dimensionless points. We assume synchronous mo-

Table 4.3: Classification error rate (%) and the corresponding average rank of each technique. Best results are in bold face. Data sets with 10 labeled points.

	<i>g241c</i>	<i>g241d</i>	<i>Digit1</i>	<i>USPS</i>	<i>COIL</i>	<i>BCI</i>	<i>Text</i>	<i>Avg. Rank</i>
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12	9.57
SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37	14.00
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	7.71
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37	10.57
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21	10.71
SGT	22.76	18.64	8.92	25.36	N/A	49.59	29.02	6.17
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	N/A	9.83
LDS	28.85	50.63	15.63	17.57	61.90	49.27	27.15	8.29
Laplacian RLS	43.95	45.68	5.44	18.99	54.54	48.97	33.68	6.00
CHM (normed)	39.03	43.01	14.86	20.53	N/A	46.90	N/A	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	7.14
Proposed Method	40.30	41.74	13.94	19.98	59.40	46.69	34.32	5.86

Table 4.4: Classification error (%) and the corresponding average rank of each technique. Best results are in bold face. Data sets with 100 labeled points.

	<i>g241c</i>	<i>g241d</i>	<i>Digit1</i>	<i>USPS</i>	<i>COIL</i>	<i>BCI</i>	<i>Text</i>	Avg. Rank
1-NN	43.93	42.45	3.89	5.81	17.35	48.67	30.11	12.57
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45	9.29
MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89	32.83	11.71
LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77	12.00
QC + CMR	22.05	28.20	3.15	6.36	10.03	46.22	25.71	7.43
Discrete Reg.	43.65	41.65	2.77	4.68	9.61	47.67	24.00	8.14
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	N/A	45.03	23.09	4.50
Cluster-Kernel	13.49	4.95	3.79	9.68	21.99	35.17	24.38	6.71
Data-Dep. Reg.	20.31	32.82	2.44	5.10	11.46	47.47	N/A	6.83
LDS	18.04	23.74	3.46	4.96	13.72	43.97	23.15	5.43
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	31.36	23.57	4.86
CHM (normed)	24.82	25.67	3.79	7.65	N/A	36.03	N/A	8.80
LGC	41.64	40.08	2.72	3.68	45.55	43.50	46.83	9.86
LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79	8.86
LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.48	12.14
Proposed Method	27.92	26.19	3.57	9.59	20.01	44.11	25.54	9.43

tion and no time delays, that is, all points move simultaneously and know the exact position of each other. The motion of unlabeled points $\mathbf{x}_i^{(u)}$ is governed by the following system:

$$\dot{\mathbf{x}}_i^{(u)}(t) = \sum_{j=1, j \neq i}^n f[\mathbf{x}_j^{(l)}(t) - \mathbf{x}_i^{(u)}(t)], i = 1, \dots, m, \quad (4.5)$$

where f is the attraction force function among instances.

As it is described by Eq. 4.5, each unlabeled instance $\mathbf{x}_i^{(u)}$ receives attractive forces from all labeled instances, and the resultant force is the sum of all individual forces. Thus, the direction and magnitude of the motion of $\mathbf{x}_i^{(u)}$ is determined by the forces applied by the labeled instances.

The attraction function is defined as a Gaussian field with parameters ϕ and β :

$$f[\mathbf{x}_j^{(l)}(t) - \mathbf{x}_i^{(u)}(t)] = [\mathbf{x}_j^{(l)}(t) - \mathbf{x}_i^{(u)}(t)] \frac{\phi}{e^{\beta \|\mathbf{x}_j^{(l)}(t) - \mathbf{x}_i^{(u)}(t)\|^2}}. \quad (4.6)$$

This attraction function is chosen in order to guarantee that the closer a point is to an attractor point, the stronger the force is. Moreover, its parameters provide an easy way to adjust the function amplitude and range, which is necessary to the correct functioning of the process. In the next subsections, the heuristics used to adjust these parameters are described in details.

4.2.2 Algorithm

In a concise form, the proposed technique can be summarized by Algorithm 10. The technique is performed iteratively in 4 steps (from 2 to 5), until all instances are labeled. The parameter initialization (step 1) is discussed in section 4.2.4.

Algorithm 10 Semi-supervised classification based on interacting forces

Input:

$\mathcal{X}^{(u)}$: unlabeled data set

$\mathcal{X}^{(l)}$: labeled data set

Output:

l_i : estimated class label for $\mathbf{x}_i^{(u)}$ ($l_i \in \mathcal{L}$)

Initialization:

1 : $(\phi, \beta, \delta) =$ initialize parameters

Classification:

repeat

2 : calculate distances among points

3 : calculate attraction forces

4 : update points' positions

5 : propagate labels

until $\mathcal{X}^{(u)} = \emptyset$

4.2.3 Stability analysis

For the sake of completeness, the stability of the system in Eq. 4.5 is analyzed by using the Lyapunov stability method (Hangos *et al.*, 2004). Based on this method, a system $f(x(t))$ is called stable if there is a candidate function $V(x) \geq 0$ positive definite, that is, $V(x) = 0$ if and only if $x = 0$, and its derivative $\dot{V}(x) = \frac{d}{dt}V(x) \leq 0$ is negative definite, that is, the equality holds if and only if $x = 0$. The problem is to find a suitable candidate function so that the above constraints are satisfied.

Given that in the proposed system the labeled instances are fixed because they do not receive any attraction force and so do not move, we turn our attention to the unlabeled points, which compose the system dynamics. Consider an unlabeled point $\mathbf{x}_i^{(u)}$ has been attracted by the resultant force function in the direction of a specific labeled point $\mathbf{x}_p^{(l)}$. In this case, $\mathbf{x}_i^{(u)}$ will putatively enter into the neighborhood of $\mathbf{x}_p^{(l)}$, δ , and become labeled. By using the difference variable $e_i(t) = \mathbf{x}_i^{(u)}(t) - \mathbf{x}_p^{(l)}$, the Lyapunov candidate function is defined as:

$$V_i = \frac{1}{2} e_i^T(t) e_i(t),$$

and its derivative is given by:

$$\dot{V}_i = \|e_i(t)\| \frac{e_i(t)^T}{\|e_i(t)\|} \dot{\mathbf{x}}_i^{(u)}(t) = e_i(t)^T \dot{\mathbf{x}}_i^{(u)}(t).$$

Substituting $\mathbf{x}_i^{(u)}$ by the expressions in Eq. 4.5 and 4.6, and dropping the time and label indexes for clarity, it results in:

$$\begin{aligned} \dot{V}_i &= -e_i^T \sum_{j=1, j \neq i}^n \frac{\phi(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} = \\ &= -e_i^T \left[\frac{\phi(\mathbf{x}_i - \mathbf{x}_p)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_p\|^2}} + \sum_{j=1, j \neq i, j \neq p}^n \frac{\phi(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \right] = \\ &= -\frac{\phi \|e_i\|^2}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_p\|^2}} + (\mathbf{x}_p - \mathbf{x}_i) \sum_{j=1, j \neq i, j \neq p}^n \frac{\phi(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \leq \\ &= -\frac{\phi \|e_i\|^2}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_p\|^2}} + \left\| (\mathbf{x}_p - \mathbf{x}_i) \sum_{j=1, j \neq i, j \neq p}^n \frac{\phi(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \right\| = \\ &= -\frac{\phi \|e_i\|^2}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_p\|^2}} + \|e_i\| \left\| \sum_{j=1, j \neq i, j \neq p}^n \frac{\phi(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \right\| = \\ &= -\phi \|e_i\| \left(\frac{\|e_i\|}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_p\|^2}} - \left\| \sum_{j=1, j \neq i, j \neq p}^n \frac{(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \right\| \right). \end{aligned}$$

Following the previously mentioned situation, $\mathbf{x}_i^{(u)}$ is closer to $\mathbf{x}_p^{(l)}$ than to all other points. Once inside the neighborhood of $\mathbf{x}_p^{(l)}$, $\mathbf{x}_i^{(u)}$ gets trapped, that is, it receives the corresponding label and do not move anymore. Therefore, one can consider that $\mathbf{x}_p^{(l)}$ is applying the strongest force over $\mathbf{x}_i^{(u)}$. So, we have:

$$\frac{\|e_i\|}{e^{\beta\|\mathbf{x}_i-\mathbf{x}_p\|^2}} \geq \left\| \sum_{j=1, j \neq i, j \neq p}^n \frac{(\mathbf{x}_i - \mathbf{x}_j)}{e^{\beta\|\mathbf{x}_i-\mathbf{x}_j\|^2}} \right\|, \quad (4.7)$$

which results in $\dot{V}_i < 0$, assuring that the system achieves a local asymptotic stable equilibrium.

4.2.4 Parameter analysis

Preventing the system to undergo some undesired situations can facilitate convergence and provide better classification accuracy. First, consider the situation when a point $\mathbf{x}_i^{(u)}$ is approximating towards a labeled attractor point $\mathbf{x}_j^{(l)}$ and getting very close to its neighborhood δ , where the attraction force is at its highest amplitude before $\mathbf{x}_i^{(u)}$ enters the neighborhood and becomes labeled. In this case, due to discretization in the simulations, it can occur that instead of entering the neighborhood, $\mathbf{x}_i^{(u)}$ overpasses it and start oscillating around $\mathbf{x}_j^{(l)}$. In this case, it is necessary to weaken the force function so that $\mathbf{x}_i^{(u)}$ take smaller steps. Other undesired situation occurs when $\mathbf{x}_i^{(u)}$ is between two opposite forces and gets stuck in a dynamical equilibrium. In this case, it is necessary to adjust the labeling neighborhood in a way that it can be possible to label $\mathbf{x}_i^{(u)}$. Hence, to avoid these situations we need to adjust the attraction function parameters ϕ , β and δ .

4.2.4.1 Parameters β and δ

The force function of Eq. 4.6 has the shape depicted in Fig. 4.4. The force source point is located at the origin of the system. It can be observed that, for a certain region, $-y^* < \text{distance} < y^*$, as an unlabeled instance approaches the labeled point, the force decreases. When $|\text{distance}| > |y^*|$, the force increases if an instance approaches the labeled point at origin.

With that in mind, consider the case in which there are two labeled points, \mathbf{x}_1 and \mathbf{x}_2 , and an unlabeled point \mathbf{x}_i between them as it is depicted in Fig. 4.5. As all labeled points apply the same attraction force, \mathbf{x}_i should converge to the nearest labeled point. However, in the specific scenario showed in Fig. 4.5, \mathbf{x}_i gets stuck in the dashed region because the attraction force by one labeled point is counterbalanced by the attraction force of the other point, characterizing a dynamical equilibrium. Moreover, as the labeling regions are smaller than the maximum force amplitude y^* , it is impossible to

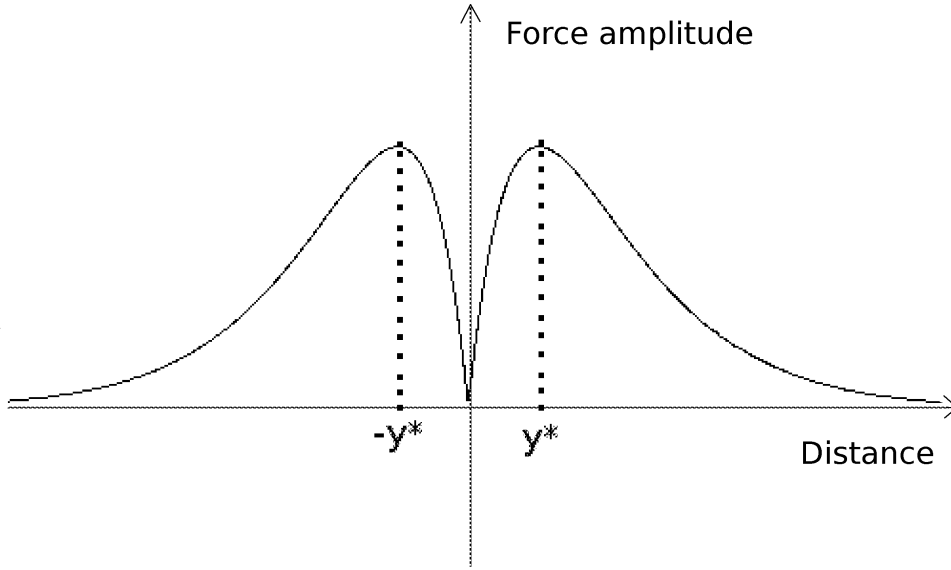


Figure 4.4: 2-dimensional visualization of the force function shape showing its maximum amplitudes at points y^* and $-y^*$. The force source point is located at the origin of the system.

assign x_i any label.

Therefore, a solution is to increase the area of the labeling region so that when x_i enters the dashed region, it receives the closest label (Fig. 4.6). We set the labeling region δ to be larger than the maximum force amplitude y^* . To accomplish that, the information of the first derivative is used as following:

$$\dot{f}(y) = \phi \frac{1 - 2\beta y^2}{e^{\beta y^2}},$$

in which, after calculating $\dot{f}(y) = 0$, y^* is found to be:

$$y^* = \frac{1}{\sqrt{2\beta}}.$$

With this result in hand, we are able to specify the region in which the technique could not converge and set the constraint $\delta > y^*$. So, the allowed combination of values for parameters β and δ is the following:

$$\delta - \frac{1}{\sqrt{2\beta}} > 0. \quad (4.8)$$

4.2.4.2 Parameter ϕ

The dynamical system in Eq. 4.5 is discretized in order to perform the simulations in this section. Such a discretization may lead the following convergence problem: an

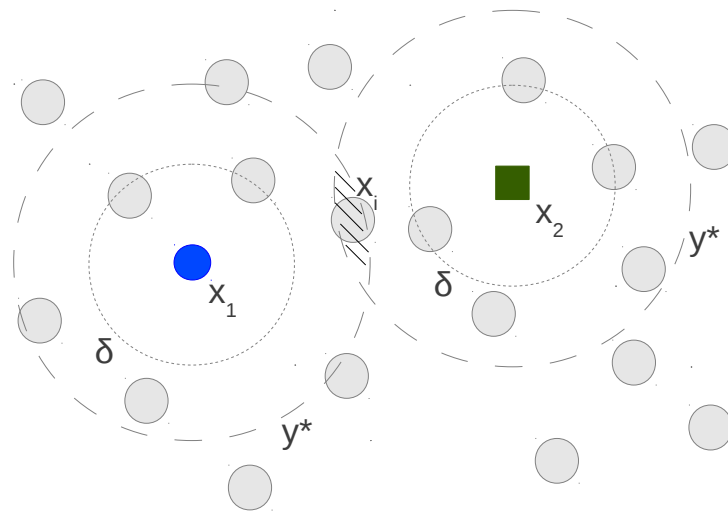


Figure 4.5: Labeling neighborhood δ is smaller than the region of maximum amplitude force γ^* .

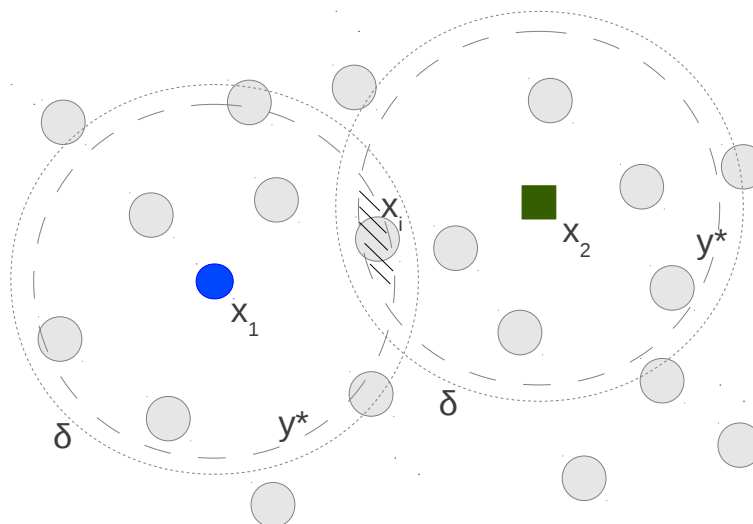


Figure 4.6: Labeling neighborhood δ is larger than the region of maximum amplitude force γ^* .

unlabeled object can come close to a labeled one, but not close enough to be labeled (outside the labeling region δ). So, the unlabeled object receives a strong force because of the small distance and, instead of entering inside the δ neighborhood, overpasses that region and starts to oscillate around the labeled object. Another undesired situation happens when all attraction forces become too weak then leading the system to demand too many steps before it reaches a solution, if it finds one. To avoid such situations, we adjust the amplitude parameter ϕ regarding the amplitude of the resultant forces applied over the unlabeled instances.

First, a precision p which scales the movements of instances at each time step is defined. The control parameter p is used in each iteration as follows:

1. Compute all forces that act over unlabeled instances (using $\phi = 1$ in Eq 4.6);
2. Calculate the resultant displacement of each unlabeled instance;
3. Normalize all displacements in such a way that the maximum displacement is equal to the precision p .

By applying this normalization, the system becomes smooth when it would be too violent, avoiding oscillations, and becomes faster when it would be too slow, requiring a smaller number of steps. In fact, p dictates the smoothness of the dynamical system: the smaller the value of p , the smoother is the system convergence.

Second, there may be another problem when classes are unbalanced. Such situation causes an undesired behavior: classes containing a large number of labeled instances apply the strongest forces, leading to wrong solutions because a unique or just a few labels propagate. Then, we have to normalize the attraction forces based on the amount of labeled instances in each class. We simply do that by dividing the force between an unlabeled point $\mathbf{x}_i^{(u)}(t)$ and a labeled point $\mathbf{x}_j^{(l)}(t)$ by the number of labeled instances of class of l ($|\mathcal{N}^{(l)}|$).

The two heuristics above can be summarized by the following:

$$\phi_i^{(l)}(t) = \frac{d_i(t)}{d_{max}(t)} \frac{p}{|\mathcal{N}^{(l)}(t)|}, \quad (4.9)$$

where $d_i(t)$ is the displacement of $\mathbf{x}_i^{(u)}(t)$ at iteration (t) , and d_{max} is the largest displacement in a time step. Thus, the parameter ϕ could be thought of as a composition of the two aforementioned normalizations: the effect of applying precision p and the normalization in terms of the amount of labeled instances. Thus ϕ changes at each time step t and may be different for labeled instances from different classes.

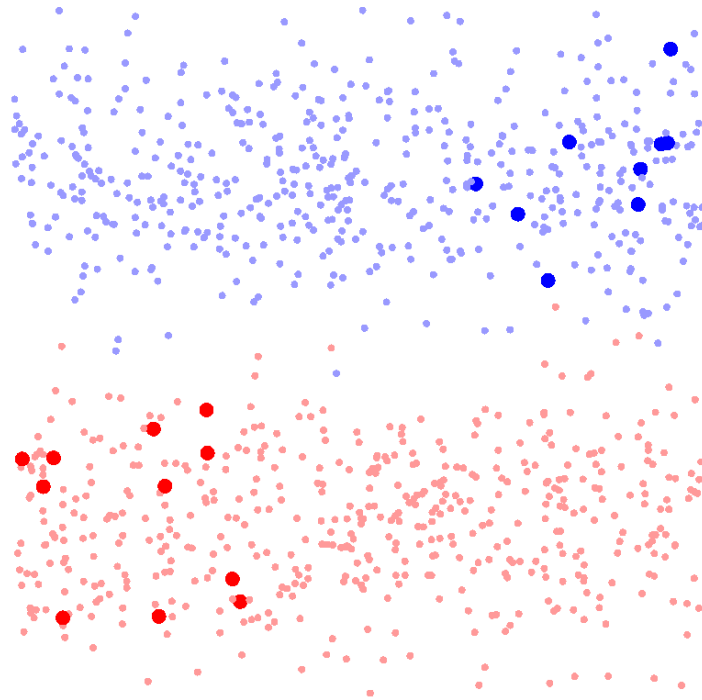


Figure 4.7: Data set composed of 2 classes, blue and red, with 500 instances each one. There are 20 initially labeled instances (darker points).

4.2.5 Experimental results

In this section, we analyze the behavior of the proposed technique on different scenarios. To conceptually illustrate the technique overall behavior, it is shown a semi-supervised example composed of 1000 instances equally divided into 2 classes, blue (upper class) and red (lower class) in Fig. 4.7. The blue class has 9 initially labeled instances, and the red class has 11. Figure 4.8 shows the classification result when applying 1-NN technique. The magenta points were incorrectly classified with the blue label and purple points were incorrectly classified with the red label. It can be seen that this technique, which relies only on the distances between the instances, can not achieve an adequate result. On the other hand, the proposed technique is capable of using the information provided by the unlabeled instances to guide the classification process, as it can be seen in Fig. 4.9, where satisfactory results were achieved.

Now, we proceed with an analysis of parameters influence. First, the influence of parameters β and δ on classification accuracy and convergence time are investigated using the data set depicted in Fig. 4.10. This data set is composed of 500 instances divided into 2 slighted mixed classes (250 instances in each class). To characterize the semi-supervised setting, only 5% of each class are initially labeled. Figure 4.11 shows the results for accuracy (left column) and convergence time (right column). Each line corresponds to different data set dimensions: 2 (Fig. 4.11(a),(b)), 10 (Fig. 4.11(c),(d)) and 50 (Fig. 4.11(e),(f)). Avoiding the region constrained by Eq. 4.7, it can be observed

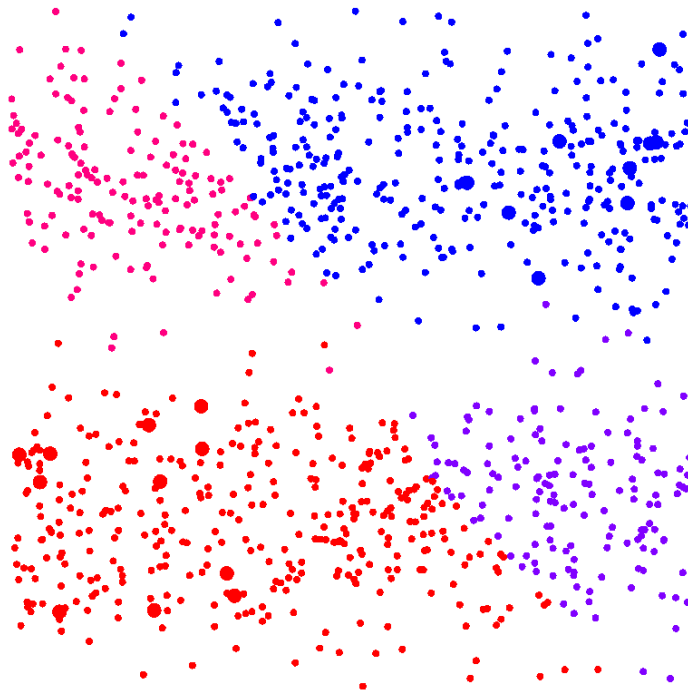


Figure 4.8: Classification result for the 1-NN technique. Magenta points are incorrectly classified with the blue label and purple points are incorrectly classified with the red label. The classification error is 31.8%.

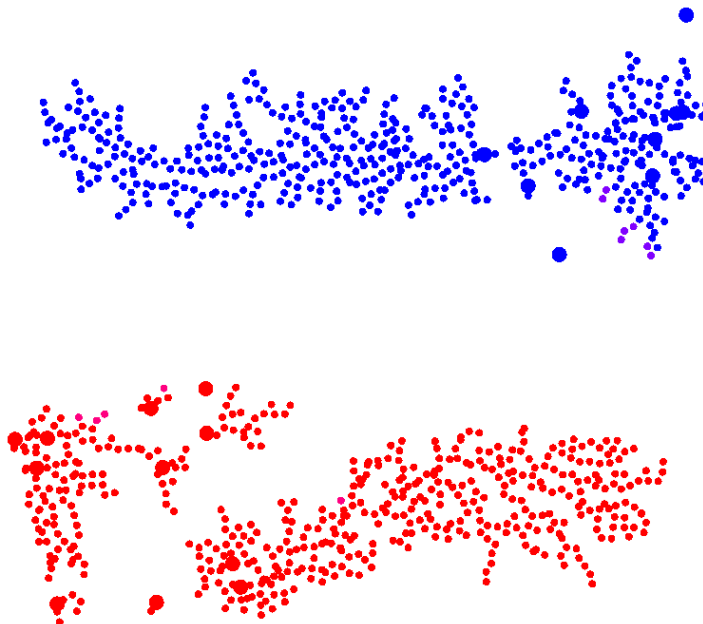


Figure 4.9: Classification result of the technique based on interacting forces. Magenta points are incorrectly classified with the blue label and purple points are incorrectly classified with the red label. The classification error is 1.2%. In this example, it can be seen that the proposed technique uses the information provided by the unlabeled instances to achieve a satisfactory classification accuracy.

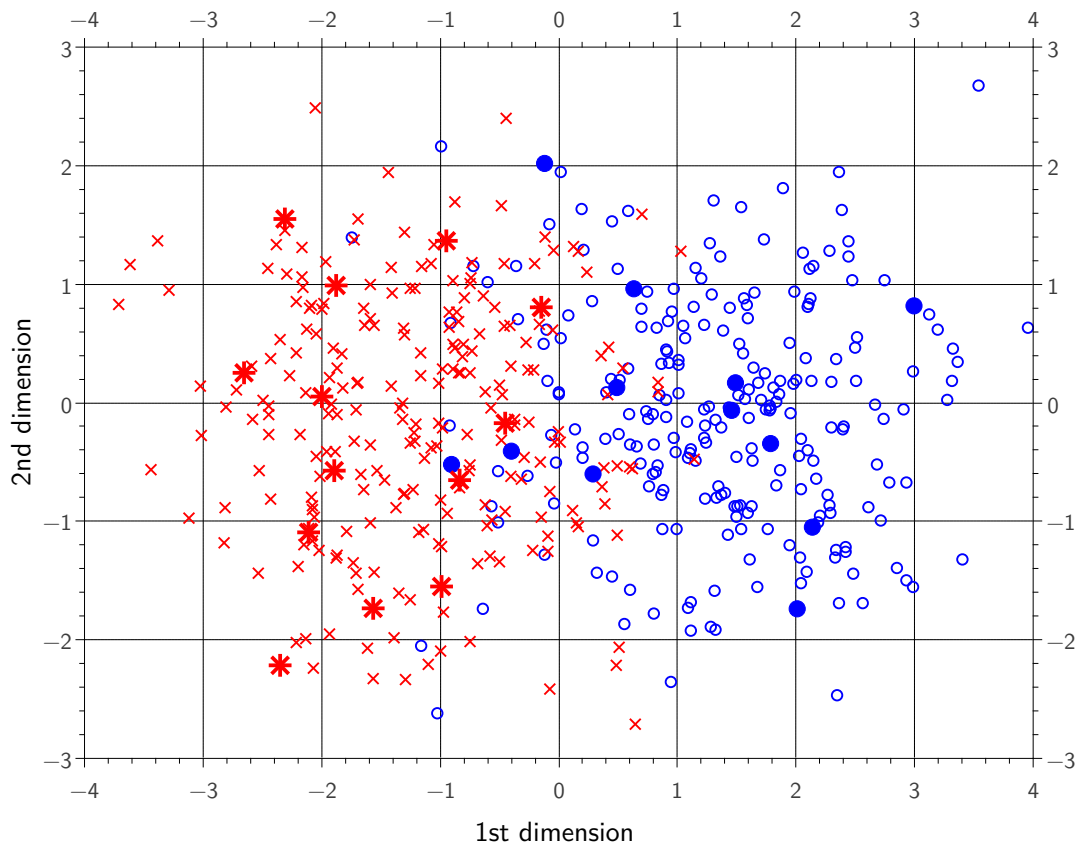


Figure 4.10: Example of a 2-dimensional artificial data set used in simulations. Small crosses and small circles represent unlabeled objects. Big stars and filled circles represent the initially labeled instances. This data set contains 500 objects (250 for each class) and 5% of them are initially labeled. Each Gaussian-class has unitary covariance and the centers are distant 2.5 units from each other along the first dimension.

that small values of parameter δ (small labeling neighborhood, when labeling is more selective) achieves slightly better accuracies. However, in this case, the convergence time is increased. This behavior is even more evident in the high-dimensional case (Fig. 4.11(e),(f)).

Furthermore, since the technique is highly dependent on the distances among data due to the dynamical movement of the instances, the data set dimension clearly affects accuracy. On the left column of Fig. 4.11, it can be observed that when the dimension increases, the accuracy decreases. For example, from a classification error of 16.9% in the 2-dimensional case, the error increased to 49.6% in the 50-dimensional case. This behavior is due to the curse of dimensionality, where an increase in the number of attributes makes the data set more sparse and decrease the variance among distances. The increase in the convergence time observed in the right column (Fig. 4.11(b), (d) and (f)) is due to the increase in number of calculations when considering a larger number of attributes.

The influence of the data set size and of the number of initially labeled instances

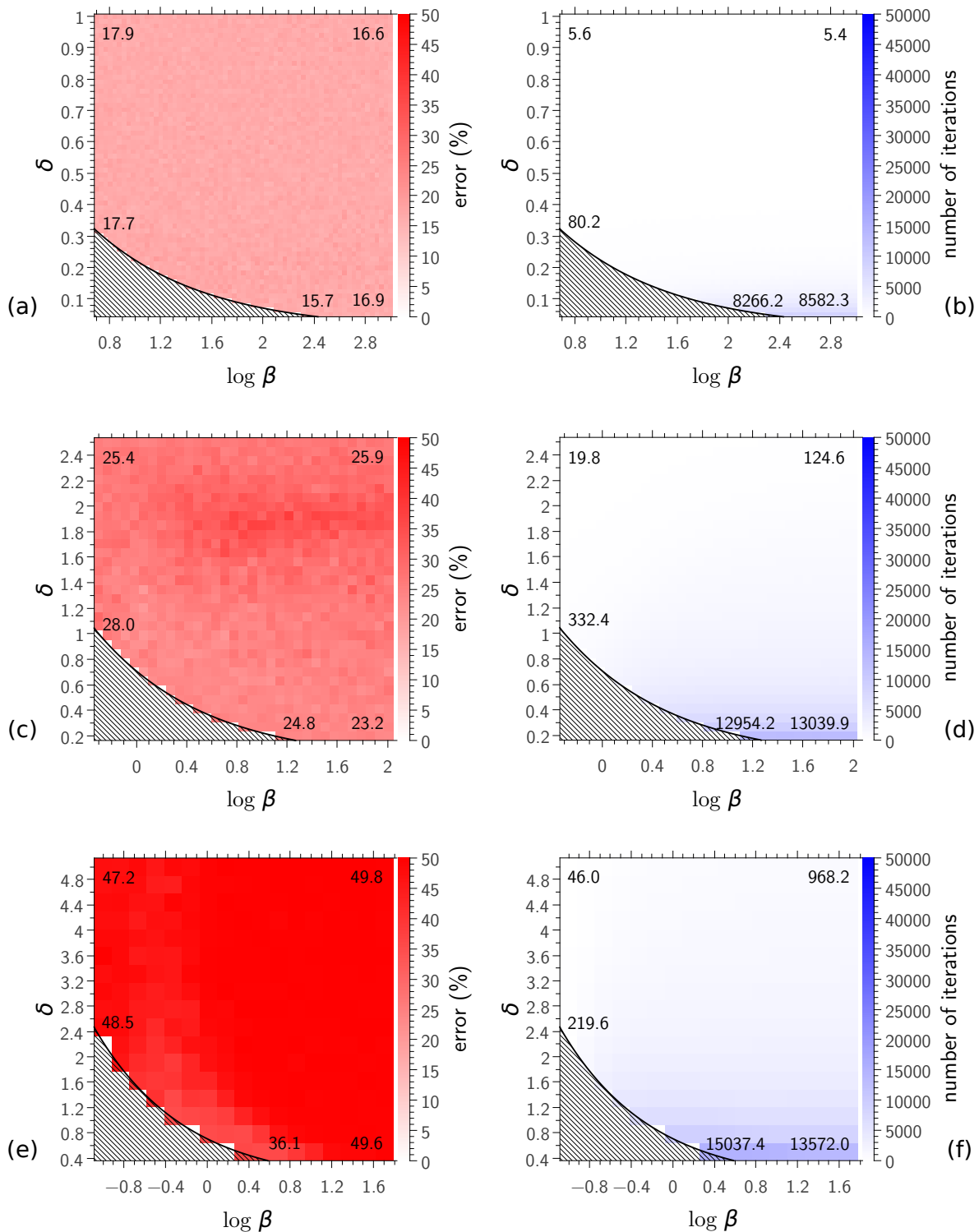


Figure 4.11: Analysis of the influence of parameters β and δ on accuracy and convergence time for Gaussian artificial data sets (see Fig. 4.10). Data set dimensions vary from (a), (b): 2; (c), (d): 10; (e), (f): 50. (a), (c), (e): classification error rate. (b), (d), (f): number of iterations to label all objects. The hatched region corresponds to $\delta - 1/\sqrt{2\beta} < 0$. Precision $p = \delta/2$. 30 simulations were averaged.

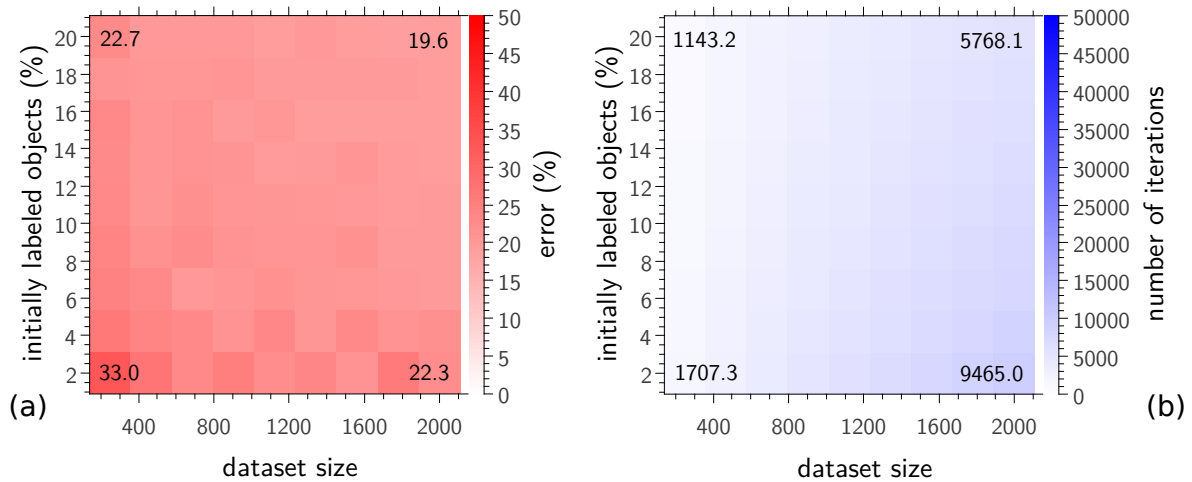


Figure 4.12: Analysis of the data set size and the number of initially labeled objects for 10-dimension Gaussian artificial data sets (see Fig. 4.10). (a): classification error rate. (b): number of iterations to label all instances. Parameters: $\beta = 5$, $\delta = 0.75$, $p = \delta/2$. 30 simulations were averaged.

were also investigated. Figure 4.12 shows the results for data set sizes increasing from 250 instances to 2000 instances, and for number of initially labeled instances increasing from 2 to 20%. As it could be expected, when the number of initially labeled instances is large, the accuracy increases (Fig. 4.12(a)). For instance, with a set size of 250, the classification error decreased from 33.0% to 22.7%, for 2 and 20 labeled instances, respectively.

Another interesting behavior can be observed also concerning the data set size. In these simulations, the larger was the size, the better was the accuracy. For example, with 2 initially labeled instances, the error decreased from 33.0% to 22.7%, for sizes of 250 and 200, respectively, an improvement of 31.2%; with 20 initially labeled instances, the error decreased from 22.7% to 19.6%, for sizes of 250 and 200, respectively, an improvement of 13.7%. This behavior is due to the fact that the information provided by the unlabeled instances is used in the classification processes in the semi-supervised technique. Hence, the larger is the number of unlabeled instances (given that the correct class representation is kept), the better should be the accuracy.

The proposed semi-supervised technique was also evaluated and compared using some benchmark data sets. Table 4.1 in subsection 4.1.3 shows a brief description of them. In order to evaluate the influence of parameters β and δ using a benchmark data set, we selected *Digit1*, which encompasses the semi-supervised assumptions. As it is evidenced in the previous simulations on artificial Gaussian data sets, in this case, a more selective neighborhood - small values for parameter δ - also increases the accuracy. In Fig. 4.13, it can be seen that for 100 initially labeled instances the error decreases from 6.9% to 5.9% for $\delta = 1$ and 0.1, respectively, an improvement of 14.5%. Figure 4.14 shows the results by using the first 5 PCA components to avoid the influ-

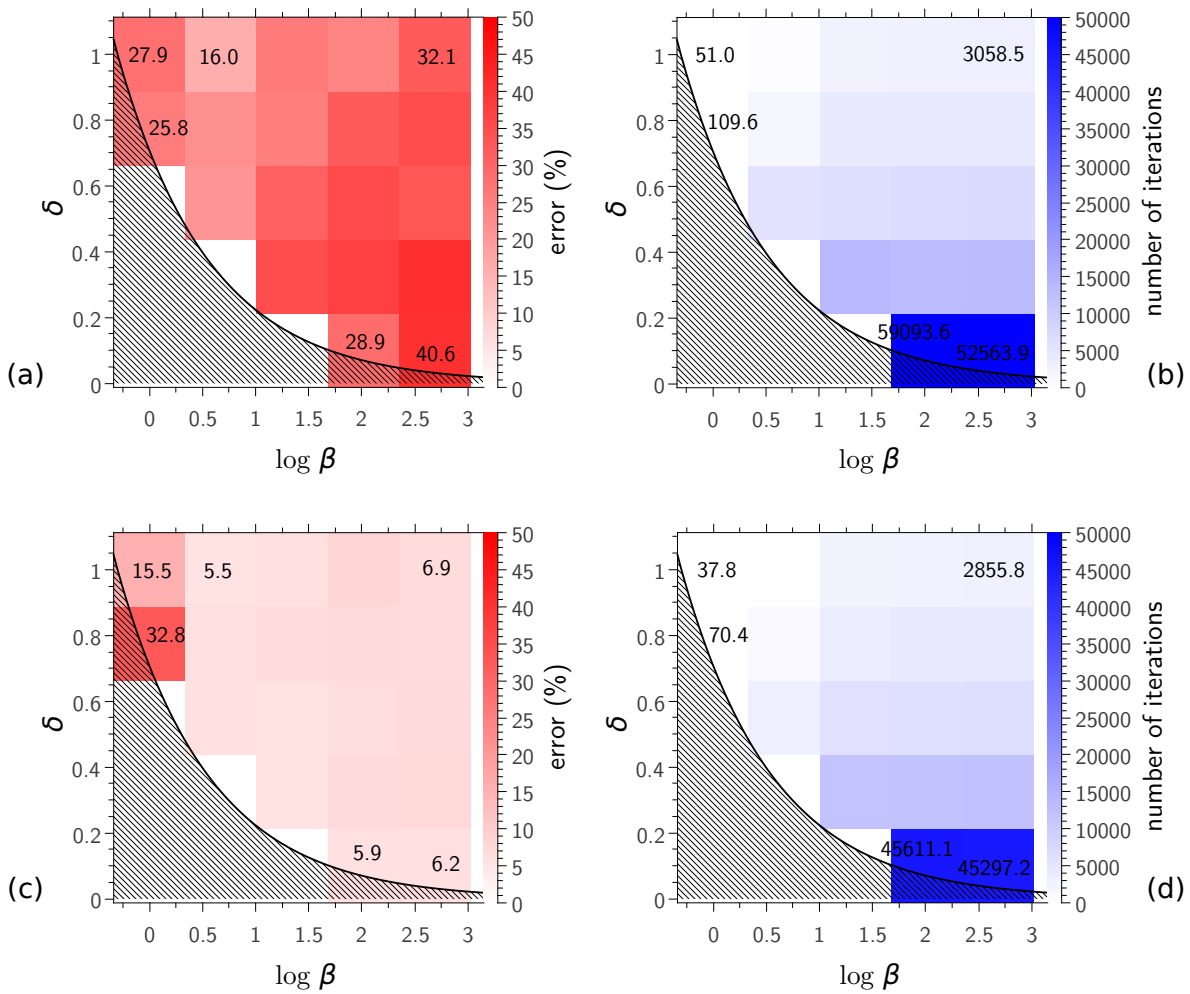


Figure 4.13: Analysis of parameters β and δ on the *Digit1* benchmark data set. (a), (c): averaged classification error rate. (b), (d): number of iterations to label all instances. (a), (b): 10 initially labeled instances. (c), (d): 100 initially labeled instances. The hatched region corresponds to $\delta - 1/\sqrt{2\beta} < 0$. Precision $p = \delta/2$. 30 simulations were averaged.

ence of dimension. In this case, when 10 initially labeled instances were used, the error decreased from 21.4% to 20.6%, for $\delta = 0.5$ and 0.05, respectively. When 100 initially labeled instances were used, the error decreased from 6.8% to 6.6%, for $\delta = 0.5$ and 0.05, respectively. The improvement in the latter case is smaller because a error of 6.8% is already very small for the data set under analysis.

We compared the proposed technique to 8 well-known and established classification techniques: k-Nearest Neighbors (k-NN), Maximum Variance Unfolding (MVU) + k-NN, Laplacian Eigenmaps (LEM) + k-NN, Support Vector Machines (SVM) with linear, Radial Basis Function - RBF, quadratic and polynomial kernels, Transductive SVM (TSVM), Spectral Graph Transducer (SGT), Low-Density Separation (LDS) and Laplacian Regularized Least Squares (LRLS). Table 4.2 in subsection 4.1.3 shows a brief description and their related references. The first 5 PCA components computed from the instances of each data set were used in all simulations and for all techniques (ex-

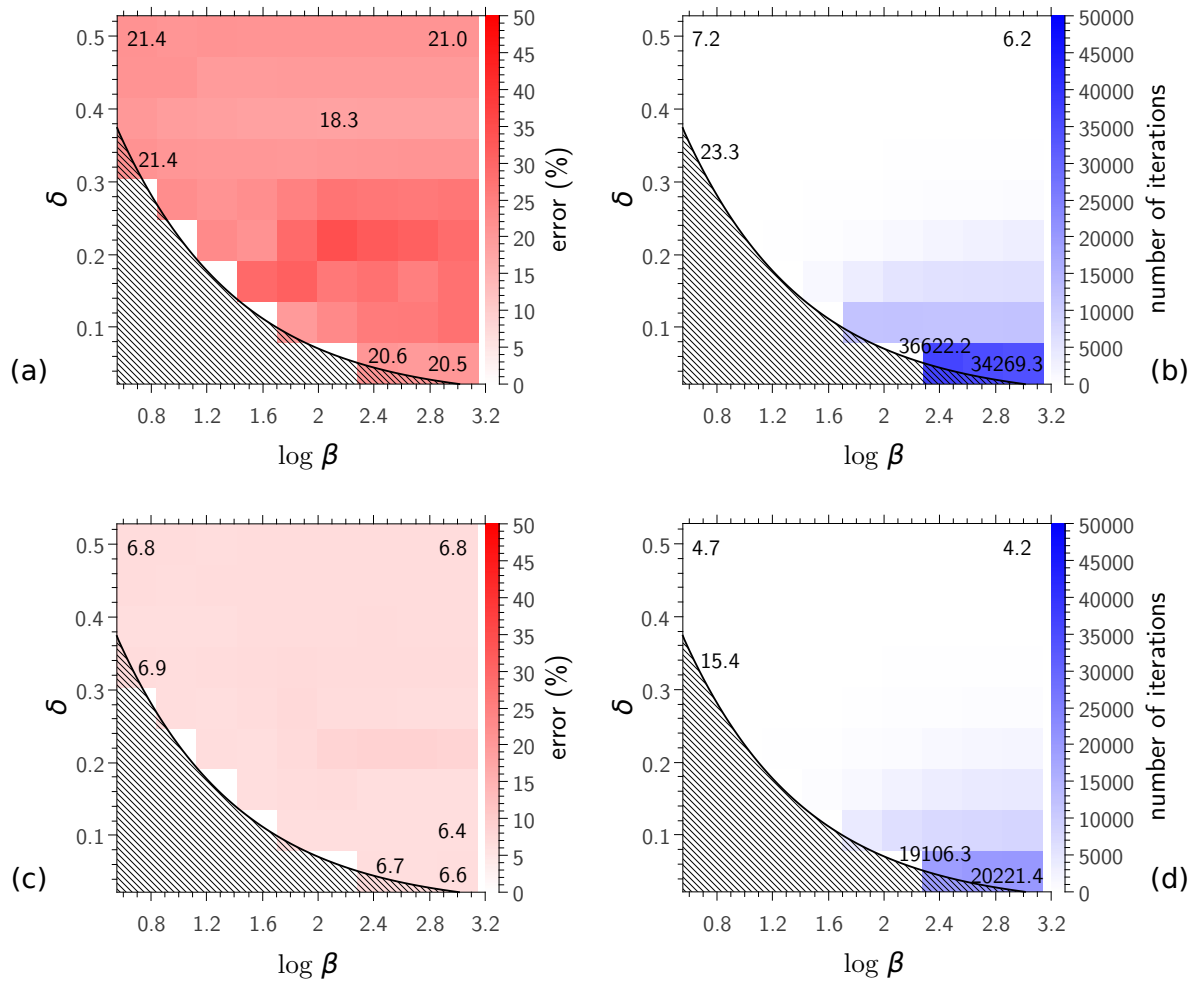


Figure 4.14: Analysis of parameters β and δ on the *Digit1* benchmark data set. (a), (c): averaged classification error rate. (b), (d): number of iterations to label all instances. (a), (b): 10 initially labeled instances. (c), (d): 100 initially labeled instances. The hatched region corresponds to $\delta - 1/\sqrt{2\beta} < 0$. Precision $p = \delta/2$. The first 5 PCA components were used to avoid the influence of data set dimension. 30 simulations were averaged.

cept for MVU and LEM, which are dimensionality reduction techniques). Due to the information loss when applying dimensionality reduction, some techniques perform better in the original feature space. However, since the proposed technique is highly sensitive to the data dimensionality, we report here the results using PCA to conduct a fair analysis.

All techniques were optimized according to their parameters. The configuration and parameter optimization for the proposed technique was done as follows. Let \bar{d} denote the average distance among all pairs of instances from the data set. We set the intervals of parameters δ ($\delta_{\min}, \delta_{\max}$) and β ($\beta_{\min}, \beta_{\max}$) as: $\delta_{\min} = \bar{d}/6$, $\delta_{\max} = \bar{d}/3$, $\beta_{\min} = 1.05 \cdot \frac{1}{2\delta_{\min}^2}$, $\beta_{\max} = 500 \cdot \beta_{\min}$.

Table 4.5 shows the results for 10 and 100 initially labeled instances, and Tables 4.6 and 4.7 show the corresponding rank. The ranking measure was calculated as follows: i) for each data set and for each technique, a rank value is determined according to its performance, that is, the best technique on a given data set is ranked as 1, the second best is ranked as 2 and so on; ii) for each technique, the final rank is averaged over all data sets. It can be observed that for 10 initially labeled instances (Table 4.6) the proposed technique resulted in the 3rd position with a rank of 5.0 ± 3.3 , preceded by LRLS (3.8 ± 2.2 - 1st) and by SGT (4.8 ± 4.3 - 2nd). For 100 initially labeled instances, the proposed technique resulted in the 4th position (5.6 ± 2.9) preceded by LRLS (3.4 ± 1.1 - 1st), LDS (4.6 ± 4.0 - 2nd) and SGT (5.2 ± 3.7 - 3rd). These results suggest that the proposed technique is at least comparable to the well-known and most competitive semi-supervised classification techniques.

Table 4.5: Predictive errors (%) on the benchmark data sets provided in (Chapelle *et al.*, 2006). The data sets were pre-processed by using the first 5 PCA components (except for MVU and LEM, which are dimensionality reduction techniques). Best results are in boldface.

	10 initially labeled instances					100 initially labeled instances				
	g241c	g241d	Digit1	USPS	BCI	g241c	g241d	Digit1	USPS	BCI
1-NN	28.95	27.20	21.77	23.93	49.10	21.40	10.98	7.04	13.60	47.36
MVU + 1-NN	47.15	45.56	14.42	23.34	47.95	43.01	38.20	2.83	6.50	47.89
LEM + 1-NN	44.05	43.22	23.47	19.82	48.74	40.28	37.49	6.12	7.64	44.83
SVM (linear)	24.00	28.67	21.53	31.52	48.50	13.34	13.99	6.79	21.60	46.92
SVM (RBF)	44.41	39.23	33.15	19.69	49.79	17.76	9.12	5.73	13.68	48.06
SVM (quad)	39.33	31.95	26.62	25.62	49.64	17.56	10.41	6.18	12.88	49.00
SVM (poly)	33.47	33.23	29.63	26.09	49.44	22.05	13.48	7.40	18.45	47.75
T SVM	21.38	31.23	20.32	30.04	48.96	13.77	21.63	9.15	14.54	45.65
SGT	17.83	12.94	14.63	26.50	49.64	14.11	6.72	4.91	13.57	48.50
LDS	17.46	36.48	18.90	23.48	49.02	13.19	8.39	4.49	19.33	47.31
Lapl. RLS	27.10	26.67	18.03	18.68	49.38	16.04	7.83	4.88	11.76	46.11
Prop. Method	27.64	27.42	18.31	18.31	49.64	21.63	10.24	6.41	12.44	45.63

Table 4.6: Rank of results for 10 initially labeled instances (see Table 4.5).

	g241c	g241d	Digit1	USPS	BCI	Average rank
1-NN	7	3	8	7	6	6.2 ± 1.9
MVU + 1-NN	12	12	1	5	1	6.2 ± 5.5
LEM + 1-NN	10	11	9	4	3	7.4 ± 3.6
SVM (linear)	4	5	7	12	2	6.0 ± 3.8
SVM (RBF)	11	10	12	3	12	9.6 ± 3.8
SVM (quad)	9	7	10	8	10	8.8 ± 1.3
SVM (poly)	8	8	11	9	8	8.8 ± 1.3
TSVM	3	6	6	11	4	6.0 ± 3.1
SGT	2	1	2	10	9	4.8 ± 4.3
LDS	1	9	5	6	5	5.2 ± 2.9
Lapl. RLS	5	2	3	2	7	3.8 ± 2.2
Prop. Method	6	4	4	1	10	5.0 ± 3.3

Table 4.7: Rank of results for 100 initially labeled instances (see Table 4.5).

	g241c	g241d	Digit1	USPS	BCI	Average rank
1-NN	8	7	10	7	7	7.8 ± 1.3
MVU + 1-NN	12	12	1	1	9	7.0 ± 5.6
LEM + 1-NN	11	11	6	2	1	6.2 ± 4.8
SVM (linear)	2	9	9	12	5	7.4 ± 3.9
SVM (RBF)	7	4	5	8	10	6.8 ± 2.4
SVM (quad)	6	6	7	5	12	7.2 ± 2.8
SVM (poly)	10	8	11	10	8	9.4 ± 1.3
TSVM	3	10	12	9	3	7.4 ± 4.2
SGT	4	1	4	6	11	5.2 ± 3.7
LDS	1	3	2	11	6	4.6 ± 4.0
Lapl. RLS	5	2	3	3	4	3.4 ± 2.9
Prop. Method	9	5	8	4	2	5.6 ± 3.3

Development of network-based techniques for unsupervised learning

The unsupervised learning paradigm is introduced in subsection 2.1.3. In this machine learning area, there is no external sources to guide the learning process, that is, information about data labels or cluster structures is unavailable. The algorithm must guide itself solely by the information extracted from the data into analysis. A particular application of this paradigm is data clustering, in which the technique must find data clusters that comprise data items more related to each other in a same cluster than to data items belonging to other clusters. Network-based techniques are a common approach because the data structures can be conveniently revealed by the underlying networks.

This chapter presents the data clustering technique introduced by Cupertino *et al.* (2013c). This network-based technique makes use of pinning control and consensus time to derive a dissimilarity measure used to cluster data. The data are represented by a connected and sparse network where nodes are dynamical elements. These elements are pinned one by one into a fixed trajectory while the remainder of the nodes try to achieve a consensus, that is, to reach the same trajectory of the pinned node. The consensus dissimilarity is the total amount of time nodes take to reach the desired trajectory. It provides a set of features for the data items and allows the detection of clusters with different shapes and sizes. An analysis on the convergence of the technique is also carried out.

5.1 Data clustering via consensus dissimilarity

In pinning control, only a small fraction of vertices in the network is expected to be controlled as follows:

$$\dot{x}_i(t) = f(x_i(t)) - c \sum_{j=1}^n l_{ij} h(x_j(t)) + u_i(t), \quad (5.1)$$

where $u_i(t) = g_i \{h[s(t) - x_i(t)]\}$ is the local feedback controller, $s(t)$ is the reference trajectory and g_i quantifies the pinning control gain for vertex i . A dissimilarity measure is derived from this concept as it is explained in the next subsections.

5.1.1 Dissimilarity measure based on consensus time

A dissimilarity measure among vertices on a network can be directly derived from the consensus time concept. Given a pinned vertex i with an arbitrary fixed trajectory, $x_i = \bar{x}$, the dissimilarity d_{ji} between vertices j and i is defined as the total amount of time it takes vertex j to reach the stationary state \bar{x} . In other words, in a discrete system, the number of time steps d_{ji} .

From Eq. 5.1, the pinning control can be reduced to the consensus problem in the presence of some pinned vertices with a fixed reference trajectory $s(t) = \bar{x}$, where \bar{x} is the desired stationary state. Such a system, with internal coupling function $h(\mathbf{x}) = \mathbf{x}$ and linear self-feedback $f[\mathbf{x}_i(t)] = \psi \mathbf{x}_i(t)$, $\psi > 0$, is defined by the following discrete-time equation:

$$\mathbf{x}_i(t+1) = \psi \mathbf{x}_i(t) - \epsilon \sum_{j=1}^n l_{ij} \mathbf{x}_j(t) + \epsilon u_i(t), \quad (5.2)$$

where $\epsilon > 0$ is the step size and $u_i(t) = g_i [\bar{x} - \mathbf{x}_i(t)]$. For the controlled vertex, $g_i = z > 0$ is the control gain; for all other vertices, $g_i = 0$.

By using the discretization procedure from Eq. (5.1) to Eq. (5.2), one must assure that the convergence is achieved within a time $t < \infty$. As stated in the previous section, the reference trajectory $s(t)$ is achieved as $t \rightarrow \infty$ for continuous systems. However, in this work, we consider the discrete system as in Eq. (5.2) and set a discretization error, for example $e = 10^{-3}$, to achieve consensus in a reasonable amount of time. This process, although can be seem too rough, is effective for clustering as it is shown in the section of experimental results.

Each vertex in a network is likely to converge to the desired state \bar{x} at different and finite times by pinning a single vertex (for example, $\mathbf{x}_1(0) = \bar{x}$ and $g_1 = z > 0$) and satisfying the parameters constraints which are further addressed in the section of convergence analysis. All other vertices must start at the same initial state: $\mathbf{x}_2(0), \mathbf{x}_3(0), \dots, \mathbf{x}_n(0) = \beta, \beta \neq \bar{x}$.

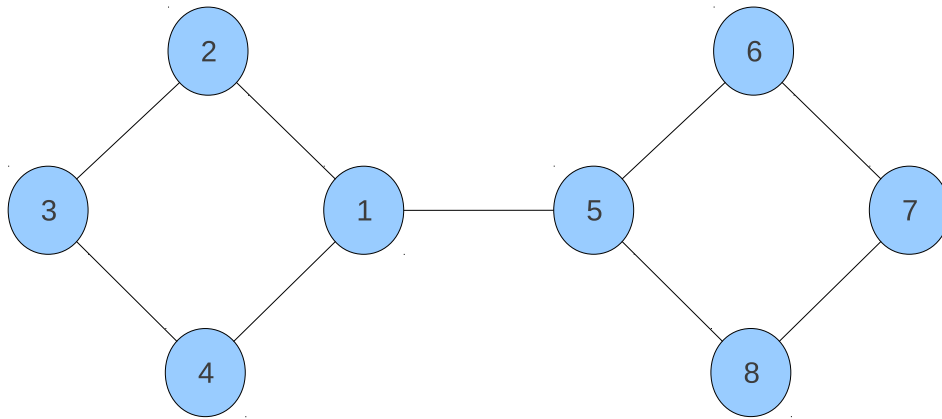


Figure 5.1: A small network composed of 8 vertices used to illustrate the calculation of the consensus dissimilarity.

As an example, consider the toy network consisting of 8 vertices depicted in Fig. 5.1. Figure 5.2 shows the state evolution for the vertices using a single pinned vertex $x_1 = \bar{x} = 0$, $\beta = 1$, $\psi = 0.05$, $\epsilon = z = 0.2$. In this simple example, it can be seen that each vertex reaches \bar{x} at different times. Specifically, vertices that lie in the same densely connected group as vertex 1 (1 to 4) represented by green and magenta lines, converge faster than the other densely connected group (5 to 8), which is represented by blue and red lines.

Figure 5.3 shows the consensus dissimilarities between some pairs of neighbor vertices. It must be noted that the calculations result in asymmetric values. In subsection 5.1.3.2, this issue is concerned because matrix D must be symmetrized before being used in the data clustering task.

5.1.2 Convergence analysis

In order to achieve the desired results when calculating the consensus time dissimilarity, the proposed system in Eq. (5.2) must be asymptotically convergent. Some works concerning the use of transversal Lyapunov exponents to prove synchronization (Li and Chen, 2003) or control (Li and Wang, 2004) of coupled chaotic maps can be found in the literature. However, the parameter's constraints to achieve an asymptotically stable behavior have showed to be too strong for discrete-time systems, hampering the pin control task (Zhang *et al.*, 2010). On the other hand, for the linear system proposed in this work, we are able to establish some specific conditions on the parameter's values in order to guarantee the desired asymptotic behavior.

Theorem 1. Consider a pinned discrete-time system with n coupled elements (in which only a single element is pinned) whose dynamics are governed by Eq. (5.2). The origin of the system

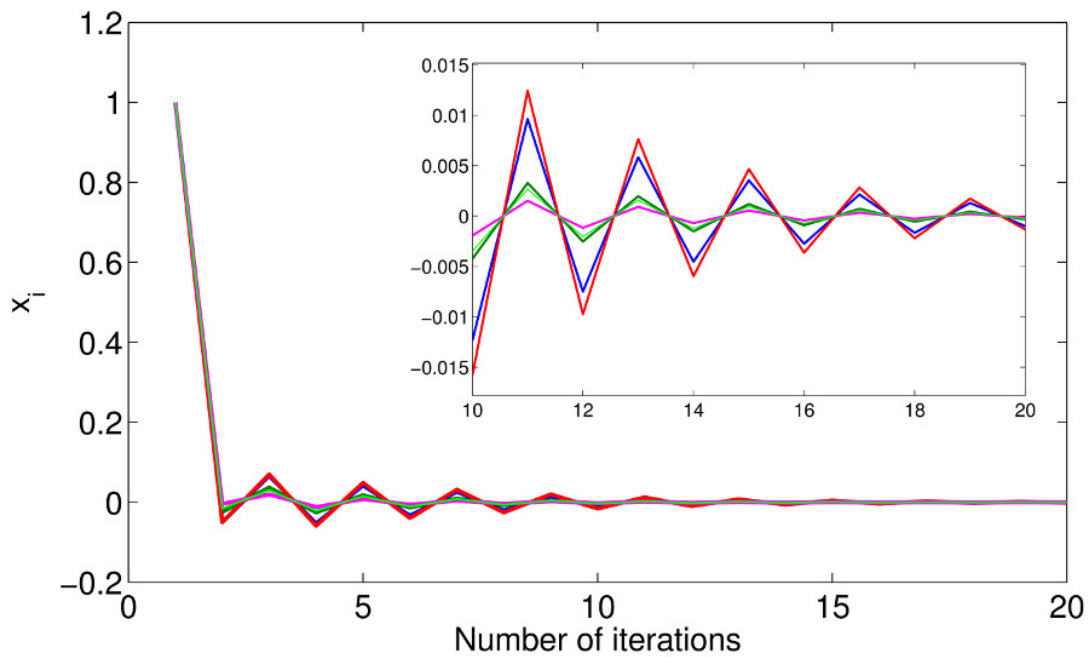


Figure 5.2: State evolution of vertices of the network depicted in Fig. 5.1. All vertices reach a consensus in the presence of a pinned vertex $x_1 = \bar{x} = 0$.

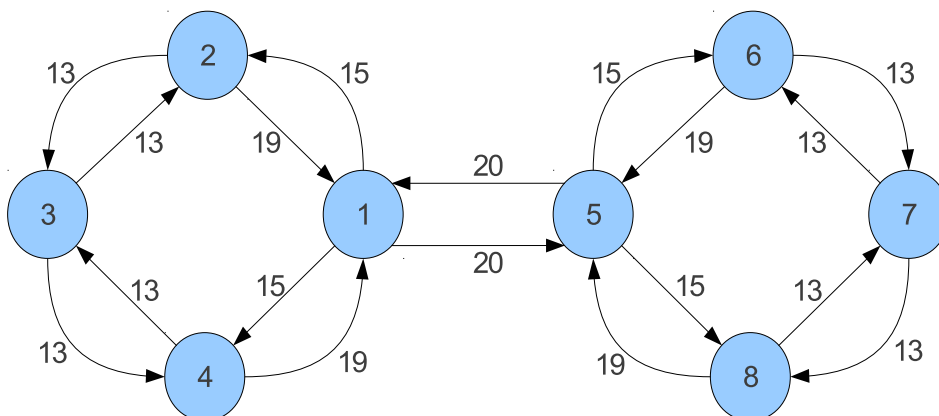


Figure 5.3: Consensus dissimilarities between some pairs of neighbor vertices of the network depicted in Fig. 5.1. The parameter setting is $x_1 = \bar{x} = 0$, $\beta = 1$, $\psi = 0.05$, $\epsilon = z = 0.2$.

is asymptotically stable for any initial condition $\mathbf{x}_i(0)$ if, and only if, the following constraints hold:

$$0 < \epsilon < \frac{C}{|l_{i,j_{\max}}|}, \quad (5.3)$$

$$\epsilon l_{i,i_{\max}} - C < \psi < \epsilon l_{i,i_{\min}} + C, \quad (5.4)$$

$$\frac{\psi}{\epsilon} - l_{i,i_{\min}} - \frac{C}{\epsilon} < z < \frac{\psi}{\epsilon} - l_{i,i_{\max}} + \frac{C}{\epsilon}, \quad (5.5)$$

in which $l_{i,j_{\max}}$ is the maximum edge weight, $l_{i,i_{\min}}$ and $l_{i,i_{\max}}$ are the smallest and the largest vertex degrees, respectively, and C is either $\sqrt{2}/n$ or $2/\sqrt{2n^2 - 1}$, for n even or odd, respectively.

Proof. The system in Eq. (5.2) can be rewritten as $X(t+1) = AX(t)$, where $X \in R^n$ is a state vector containing all vertices, and $A \in R^{n \times n}$ is a symmetric matrix of the following form:

$$A = \begin{bmatrix} \psi - \epsilon(l_{1,1} + z) & \epsilon l_{1,2} & \dots & \epsilon l_{1,n} \\ \epsilon l_{2,1} & \psi - \epsilon l_{2,2} & \dots & \epsilon l_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon l_{n,1} & \epsilon l_{n,2} & \dots & \psi - \epsilon l_{n,n} \end{bmatrix}, \quad (5.6)$$

in which the pinned vertex was given index 1 without loss of generality.

The origin of the discrete system in Eq. (5.2) is asymptotically stable, that is, the system converges to zero, if, and only if, A is stable or, in other words, if, and only if, $r(A) < 1$, where $r(A) = \lambda_{i_{\max}}$ is the spectral radius of A (LaSalle, 1986).

The spectral radius of matrix A can be limited if the matrix entries are constrained in some interval $[a, b]$. Let A be a real and symmetric $n \times n$ matrix, $n \geq 2$, with entries lying on the interval $[-a, a]$, $a > 0$. Being λ_{\max} and λ_{\min} the largest and the smallest eigenvalues, respectively, and $s(A) = \lambda_{\max} - \lambda_{\min}$, the spread of A , then, when $s(A)$ is symmetric to the origin (Zhan, 2005),

$$s(A) \leq \begin{cases} an\sqrt{2} & \text{if } n \text{ is even,} \\ a\sqrt{2n^2 - 1} & \text{if } n \text{ is odd.} \end{cases} \quad (5.7)$$

Therefore, the spectral radius of A can be limited to $r(A) < 1$ by doing $\lambda_{\max} < 1$ and $\lambda_{\min} > -1$, that is, $s(A) < 2$.

So, to guarantee $r(A) < 1$, the condition of Eq. 5.7 is applied to confine each element of the matrix in Eq. 5.6. Considering the case in which n is even ($s(A) \leq an\sqrt{2}$), and using $C = \sqrt{2}/n$, the following results are found for each matrix entry:

For $a = \epsilon l_{i,j}$:

$$\begin{aligned} \epsilon l_{i,j} n \sqrt{2} &< 2 \\ \epsilon &< \frac{C}{|l_{i,j_{max}}|}, \end{aligned}$$

retrieving Eq. 5.3.

For $a = \psi - \epsilon l_{i,i}$:

$$\begin{aligned} (\psi - \epsilon l_{i,i}) n \sqrt{2} &< 2 \\ \psi &< \epsilon l_{i,i_{min}} + C, \end{aligned}$$

and the negative case:

$$\begin{aligned} (\psi - \epsilon l_{i,i}) n \sqrt{2} &> -2 \\ \psi &> \epsilon l_{i,i_{max}} - C, \end{aligned}$$

retrieving Eq. 5.4.

For $a = \psi - \epsilon(l_{i,i} + z)$:

$$\begin{aligned} (\psi - \epsilon(l_{i,i} + z)) n \sqrt{2} &< 2 \\ z &> \frac{\psi}{\epsilon} - l_{i,i_{min}} - \frac{C}{\epsilon}, \end{aligned}$$

and the negative case:

$$\begin{aligned} (\psi - \epsilon(l_{i,i} + z)) n \sqrt{2} &> -2 \\ z &< \frac{\psi}{\epsilon} - l_{i,i_{max}} + \frac{C}{\epsilon}, \end{aligned}$$

retrieving Eq. 5.5.

Equivalently, the same results hold for $C = 2/\sqrt{2n^2 - 1}$ when n is odd. \square

Remark 4. The above theorem gives sufficient conditions for the discrete-time system in Eq. (5.2) to be convergent. However, the implications of different possible values to the parameters when applying the system to compute the consensus time dissimilarities and performing data clustering should be discussed. Parameter ϵ is responsible for the speed of convergence: the greater its value, the faster is the convergence of the system. If the system converges too fast, all dissimilarities between all pairs of vertices may be very similar. In this case, the consensus measure will be inappropriate for clustering, once in the clustering task one expects low dissimilarity between close vertices and high dissimilarity between far vertices. Parameter ψ acts in the same way. When applying this method, convenient values should be chosen for these

parameters accordingly to the intervals of convergence. In all simulations performed in this work, the parameters' values are chosen to achieve high clustering accuracies, that is, by setting small values for ϵ and ψ while, at the same time, setting relative large values for parameter z to perform a strong pinning control.

As an illustrative example, for the network depicted in Fig. 5.1, where $l_{i,i_{max}} = 0.28$, $l_{i,i_{min}} = 0.22$ and $l_{i,j_{max}} = 0.05$, the following limits have been found:

$$\begin{aligned} 0 < \epsilon < 3.54, \\ -0.12 < \psi < 0.22, & \quad \text{for } \epsilon = 0.2, \\ -0.85 < z < 0.85, & \quad \text{for } \psi = 0.05. \end{aligned}$$

5.1.3 Technique description

This subsection introduces the network-based data clustering technique by using the previously explained consensus time dissimilarity. The technique consists of two main steps:

1. Building a network from the original data set by using the method described in subsection 5.1.3.1;
2. Detection of data clusters on the constructed network as described in subsection 5.1.3.2.

5.1.3.1 Network construction

A data set can be mapped into a network by using one of the following methods: i) each vertex, representing a data item, is connected to its k -nearest neighbors (the k -most similar data items); ii) each vertex is connected to all vertices within a predefined distance; iii) more similar vertices have higher probability to be connected than fewer similar vertices. However the drawback of these methods is that they may construct either disconnected or densely connected networks. As a consequence, they cannot be suitable for reproducing reliable network clusters that correspond to the expected data clusters. In order to overcome this problem, we propose a method of network construction based on the Single-Linkage (SL) clustering heuristic (Sibson, 1973) that is capable of constructing connected and sparse networks which, at the same time, tend to keep the cluster structure of the original data set. The steps of the proposed algorithm are described as follows:

¹According to the SL heuristic, the dissimilarity between two groups is computed as the dissimilarity between the two closest vertices that connect both groups.

²The threshold value is based on the assumption that vertices of the same group generally form a uniform density of dissimilarities. If no pair satisfies the threshold condition, an edge is created between

Algorithm 11 Network construction based on the single-linkage clustering heuristic**Input:** \mathcal{X} : unlabeled data set**Output:** \mathcal{N} : clustered and sparse network**Parameters:** k : number of node pairs to connect γ : threshold value**Network construction:**1 : start with each vertex i into a different group G_i (n initial groups)

2 : calculate the distances among all groups by using a distance measure, for example, the Euclidean distance

repeat3 : find the two closest groups¹ and denote them by G_1 and G_2 4 : calculate the average dissimilarity among vertices inside each group G_1 and G_2 , and denote them by d_1 and d_2 , respectively5 : select the k -most similar pairs of vertices that connect G_1 and G_2 , and create an edge between each selected pair if its dissimilarity is smaller than a threshold² defined as $d_{thr} = \gamma \max(d_1, d_2)$, where $\gamma > 0$. This step joins G_1 and G_2 into a larger group

6 : update the adjacency matrix by calculating the dissimilarities between the group formed in step 6 and all other groups

until there is a single group of vertices

As an example of the application of the Alg. 11 for network construction, Fig. 5.4 shows the results for an artificial data set composed of 3 clusters of different sizes and densities. In this example, 3 different values for parameter k are used: $k = 3, 5$ and 20. It can be observed that for all cases it results in a connected network with a fair distribution of links among clusters, that is, the connections inside the clusters are dense while the connections inter-clusters are sparse.

Remark 5. As it has been mentioned, the Alg. 11 for network construction uses a combination of the SL and kNN heuristics. These specific choices have two main purposes. First, the SL heuristic states that the distance between two groups of vertices is given by the distance between their two closest vertices. By doing so, it is guaranteed that when two groups are being connected, the connection starts by the closest vertices, that is, the border vertices of each group. As it is reasonable to consider border vertices as a natural division or frontier among different groups, we assume that, when joining different groups, instead of connecting far away vertices, it is better to connect border vertices. Second, it is necessary to define how many vertices will be connected. As in most common clustering cases one may not have the data distribution for each cluster, nor even the real number of clusters, we use a parameter k to set the number of

the most similar pair to guarantee a connected network. A connected network is necessary for the convergence of all vertices when the dissimilarity measure is computed by using the consensus time dissimilarity as it is explained in section 5.1.1.

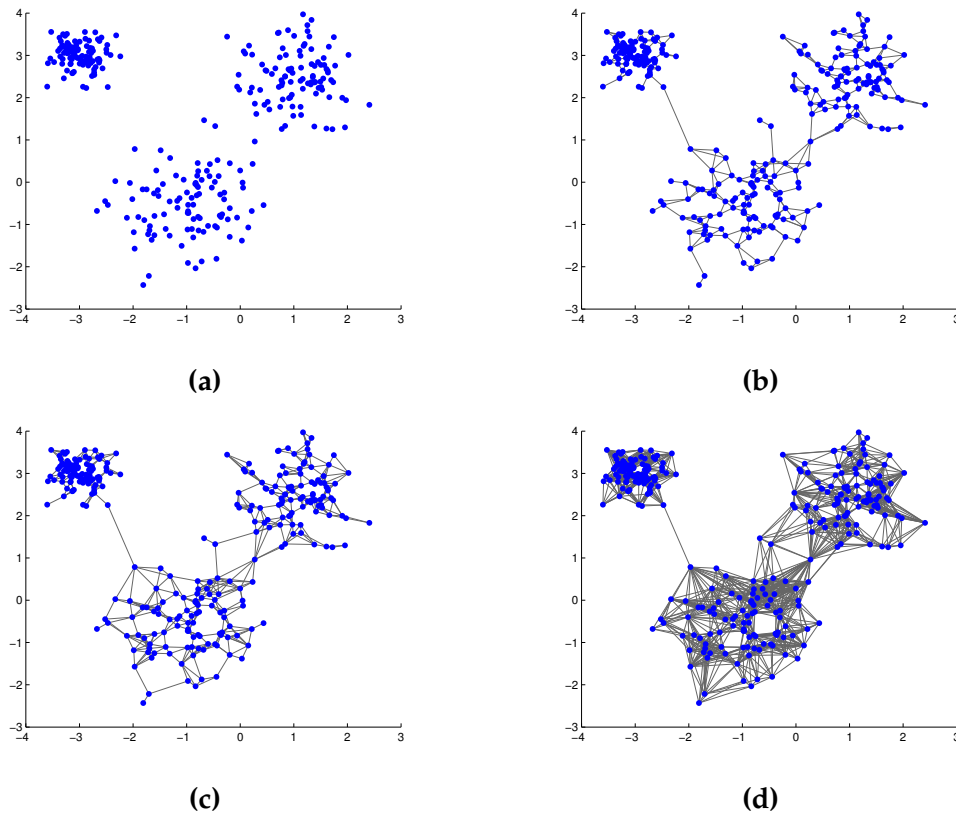


Figure 5.4: Networks constructed by using the Alg. 11 from the artificial data set in (a). The value of parameter k is (b) $k = 3$, (c) $k = 5$ and (d) $k = 20$. In all cases, $\gamma = 3$.

vertices to be connected. As it can be seen in Fig. 5.4, k is responsible for strengthening the intra-cluster connections, while keeping inter-cluster connections sparse. Thus, parameter k is used for model selection.

5.1.3.2 Cluster detection

After creating a network by applying the Alg. 11 of the previous subsection, data clusters can be detected. First, the consensus time dissimilarity (section 5.1.1) is calculated for the given network to construct a dissimilarity matrix D . The basic idea is that, for the same cluster, it takes vertices similar time to reach a common state, resulting in similar values in D . Thus, applying a clustering algorithm in D , a hierarchical structure of clusters is obtained. For this purpose, matrix D must be transformed into a symmetric matrix because most hierarchical data clustering techniques need a symmetric dissimilarity matrix as input. A simple approach to do that is to take the average dissimilarity between d_{ij} and d_{ji} . Therefore the symmetric dissimilarity matrix D_s is defined as:

$$D_s = (D + D^T)/2. \quad (5.8)$$

In a brief, the steps to detect clusters by using the proposed dissimilarity measure are described as follows:

1. Calculate the asymmetric dissimilarity matrix D as described in section 5.1.1;
2. Transform D into D_s , according to Eq. (5.8);
3. Apply a hierarchical clustering method on D_s ;
4. Choose a partition from the dendrogram.

5.1.4 Experimental results

The effectiveness of the proposed clustering technique has been evaluated by simulating 13 real and diverse data sets selected from the UCI machine learning repository as shown in Table 3.1 (Bache and Lichman, 2013). As can be seen in this table, the selection encompassed diversity on data domains as well as considered different number of classes, attributes and set sizes (they vary from 3 to 15, 4 to 91 and 132 to 2310, respectively). These data sets contain multi-cluster data, and many of them present complex data distribution with high mixing among clusters. In the simulations, eventual categorical attributes, such as in Balance data set, were treated as numerical.

Before proceeding to parameter analysis and comparison to other techniques, different techniques for hierarchical clustering were tested in the step 3 of subsection 5.1.3.2. To cluster data on matrix D_s , we tested many hierarchical techniques such as SL, Weighted-Linkage (WL) (Jain and Dubes, 1988), Average-Linkage (AL) (Chehreghani *et al.*, 2008) and Complete-Linkage (CL) (Boberg and Salakoski, 1997). The accuracy was measured by using the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985; Vinh *et al.*, 2009). Basically, the ARI is the Rand index (Rand, 1971) corrected for chance. Given two clusterings, say U and V , it take values n_{ij} from a contingency table, where n_{ij} denotes the number of instances common to clusters U_i and V_j . It has the following form:

$$Adjusted_Index = \frac{Index - Expected_Index}{Max_Index - Expected_Index}$$

or, specifically,

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}$$

where a_i and b_j are marginal sums, and n is the total number of instances. To retrieve the clusters from the dendrograms, we have cut at the smallest height at which a horizontal cut through the tree left C or fewer clusters, being C the number of clusters known in advance. From Table 5.1, it can be seen that the AL resulted in the best clustering accuracies for the proposed technique.

Table 5.1: Clustering accuracy of the proposed technique using different hierarchical clustering methods. The value in parenthesis shows the optimal value of parameter k used in network construction. The best results are in boldface. Last line shows the averaged results followed by the standard deviation.

Data set	Consensus+WL	Consensus+AL	Consensus+CL	Consensus+SL
Hayes-Hoth	0.10 (22)	0.10 (5)	-0.01 (1)	0.02 (11)
Iris	0.88 (23)	0.94 (1)	0.04 (29)	0.57 (13)
Teaching	0.04 (2)	0.03 (1)	0.00 (3)	0.00 (1)
Wine	0.42 (23)	0.43 (1)	0.04 (30)	0.12 (1)
Image	0.48 (15)	0.51 (3)	0.03 (30)	0.11 (1)
Glass	0.27 (20)	0.26 (3)	0.04 (10)	0.21 (2)
E. Coli	0.60 (10)	0.51 (6)	0.02 (4)	0.04 (1)
Libras	0.39 (11)	0.41 (1)	0.03 (27)	0.06 (2)
Balance	0.11 (7)	0.18 (4)	0.00 (1)	0.00 (1)
Vowel	0.22 (16)	0.21 (2)	0.00 (30)	0.01 (16)
Yeast	0.17 (21)	0.17 (2)	0.00 (23)	0.00 (23)
Wine Q. Red	0.03 (2)	0.02 (1)	0.00 (1)	0.00 (1)
Segment	0.58 (27)	0.62 (1)	0.00 (30)	0.01 (30)
Average	0.33 ± 0.26	0.34 ± 0.26	0.02 ± 0.02	0.09 ± 0.16

5.1.4.1 Influence of parameter k

The influence of parameter k for network construction, in step 7 of Alg. 11, was analyzed. As has been stated before, parameter k is used for model selection as one may not know *a priori* the distribution of the data set he/she is dealing with. To analyze its influence on the proposed technique, we simulated the data sets in Table 3.1 by optimizing k in the set $[1, 2, \dots, 30]$ by the grid method. The hierarchical technique used for clustering (step 3 of subsection 5.1.3.2) was the AL. In all simulations, $\bar{x} = 0$, $\beta = 1$, $\gamma = 3$, $\epsilon = 0.0025$, $\psi = 0.02$ and $z = 0.2$.

We compared the optimal values of parameter k to the cluster distribution of each data set. The data set dispersion was calculated in terms of the Fisher ratio: the ratio of the between-cluster scatter to the within-cluster scatter in the feature space. The larger the Fisher ratio, the more well-defined are the clusters. The optimal value for parameter k was taken from the above mentioned interval that resulted in the largest ARI value.

Figure 5.5 shows the normalized results in the interval $[0, 1]$. It can be noted that there is a high dependency between the cluster dispersion and the optimal value for parameter k . Actually, the Pearson's correlation coefficient between both curves is 0.60. This measure lies in the interval $[-1, 1]$, being -1 a total anticorrelation and 1 a perfect linear dependency. Therefore, the value 0.60 means a high correlation between the k value and the data set cluster dispersion. In other words, when clusters are separated, the usage of a high value for parameter k can construct networks with well-defined clusters, that is, a large number of connections inside clusters and sparse connections among different clusters. On the other hand, when clusters are less separated or present a mixture among them, the usage of a small value for parameter k can alleviate the construction of networks with high mixing among clusters.

5.1.4.2 Comparisons

Eight well-known clustering techniques were simulated for comparison purposes. Five of them are traditional techniques: AL, WL, CL, SL and k -means (Chiang *et al.*, 2011; Linde and Buzo, 1980); the other three are based on eigenanalysis of graph-Laplacian: unnormalized spectral clustering (Spec1) (Luxburg, 2007), normalized spectral clustering according to Shi and Malik (Spec2) (Shi and Malik, 2000) and normalized spectral clustering according to Ng *et al.* (Spec3) (Ng *et al.*, 2001).

In all experiments, the proposed technique was simulated with the following parameter setting: $\bar{x} = 0$, $\beta = 1$, $\epsilon = 0.0025$, $\psi = 0.02$ and $z = 0.2$. Networks were constructed using the method described in subsection 5.1.3.1 with parameters $\gamma = 3$ and k varying from 1 to 30. We reported here the results using AL, which resulted the best accuracies. For the k -means technique, the results were averaged over 100 simu-

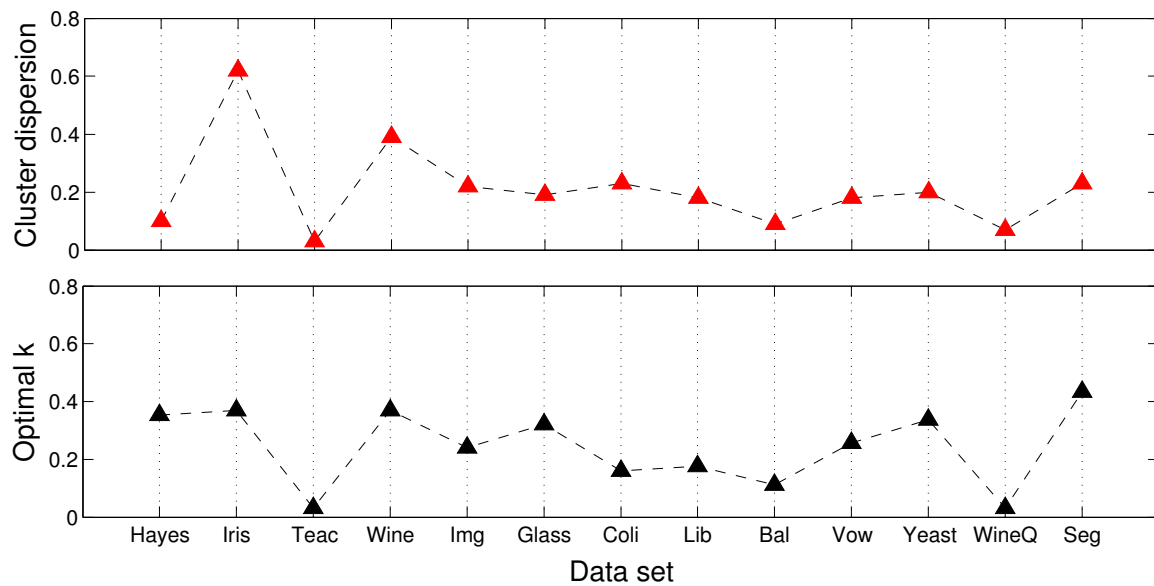


Figure 5.5: Relation of parameter k to the data set cluster dispersion. The Pearson's correlation coefficient between both curves is 0.60, indicating a high linear dependency between the two values.

lations and each simulation was optimized by selecting randomly and uniformly 10% of data to the initial seed setting. For the spectral techniques, the Alg. 11 for network construction in subsection 5.1.3.1 was applied.

Table 5.2 shows the results in terms of ARI with the optimal value of parameter k for model selection. Table 5.3 shows the rank of the techniques for each data set. To compute the rank, the accuracy achieved by a technique in a data set is ranked among all other methods for the same set. The averaged rank over all data sets can be found at the last row. The smaller the average rank, the better were the accuracies achieved by the technique. In the tested data sets, the proposed clustering method achieved the best rank (2.38), followed by Spec2 (2.92) and Spec3 (3.54).

Table 5.2: Clustering accuracy calculated by the ARI. The value in parenthesis shows the optimal value of parameter k used in network construction. For k -means it is showed the averaged results over 100 simulations followed by the standard deviation. The best results are in boldface.

Data set	Consensus+AL	Spec1+AL	Spec2+AL	Spec3+AL	k-means	WL	AL	SL	CL
Hayes-Hoth	0.10 (1)	0.10 (10)	0.10 (23)	0.10 (15)	0.10 ± 0.07	0.20	0.08	0.01	-0.01
Iris	0.94 (11)	0.88 (27)	0.90 (4)	0.90 (3)	0.68 ± 0.11	0.75	0.76	0.56	0.64
Teaching	0.03 (2)	0.01 (7)	0.01 (7)	0.01 (5)	0.00 ± 0.01	0.01	0.01	0.00	0.01
Wine	0.43 (19)	0.38 (10)	0.40 (23)	0.41 (7)	0.36 ± 0.01	0.32	0.29	0.01	0.37
Image	0.51 (5)	0.42 (5)	0.53 (12)	0.55 (17)	0.35 ± 0.05	0.19	0.18	0.10	0.17
Glass	0.26 (3)	0.26 (3)	0.28 (22)	0.28 (28)	0.26 ± 0.03	0.05	0.02	0.01	0.23
E. Coli	0.51 (7)	0.68 (6)	0.67 (3)	0.42 (3)	0.42 ± 0.07	0.68	0.74	0.04	0.62
Libras	0.41 (6)	0.36 (4)	0.38 (4)	0.39 (40)	0.30 ± 0.02	0.30	0.29	0.00	0.23
Balance	0.18 (3)	0.18 (4)	0.18 (4)	0.13 (4)	0.14 ± 0.04	0.08	0.19	0.00	0.14
Vowel	0.21 (3)	0.19 (14)	0.19 (4)	0.19 (10)	0.21 ± 0.01	0.22	0.16	0.00	0.18
Yeast	0.17 (13)	0.12 (1)	0.23 (4)	0.16 (6)	0.14 ± 0.01	0.11	0.01	0.01	0.08
Wine Q. Red	0.03 (1)	0.02 (7)	0.02 (1)	0.02 (2)	0.00 ± 0.00	-0.02	0.01	0.00	0.00
Segment	0.62 (21)	0.39 (11)	0.47 (24)	0.55 (1)	0.36 ± 0.05	0.00	0.00	0.00	0.10

Table 5.3: Rank of techniques for the results in Table 5.2. The best results are in boldface.

Data set	Consensus+AL	Spec1+AL	Spec2+AL	Spec3+AL	k-means	WL	AL	SL	CL
Hayes-Hoth	5	3	2	5	4	1	7	8	9
Iris	1	4	2	2	7	6	5	9	8
Teaching	1	5	5	4	8	7	2	9	3
Wine	1	4	3	2	6	7	8	9	5
Image	3	4	2	1	5	6	7	9	8
Glass	3	4	1	2	5	7	8	9	6
E. Coli	6	2	4	8	7	3	1	9	5
Libras	1	4	3	2	6	5	7	9	8
Balance	4	2	2	7	5	8	1	9	6
Vowel	2	4	6	5	3	1	8	9	7
Yeast	2	5	1	3	4	6	8	9	7
Wine Q. Red	1	2	4	3	7	9	5	6	8
Segment	1	4	3	2	5	7	9	7	6
Avg. rank	2.38	3.62	2.92	3.54	5.54	5.62	5.85	8.54	6.62

Conclusions

The results reported in this thesis have joined three main research areas: machine learning, complex networks and dynamical processes. The main goal has been to enhance and develop machine learning techniques by exploring the advantages of network representation and dynamical processes on networks. The motivations came from the fact that network representation unifies the structure, dynamics and functions of the system it represents, and it is able to evidence topological structures. These characteristics make networks suitable for machine learning tasks by revealing intrinsic structures and their evolutions within data set relationships. For instance, it is possible to perform classification tasks by using the heuristic of ease of access as it is described in this thesis. This method differs from traditional classification in which decision borders are defined to separate groups in the attribute space. In another way, the method uses random walk in the network where classes are identified by network connections between data instances. More interestingly, a high-level approach for classification has been explored. In this paradigm, the data are not classified by using only its physical attributes, but also using the semantic structure formed by the data in a network. In this case, even if different classes are overlapped in the attribute space, they can be correctly identified by taking into account the semantic structure each class forms. The network representation feature has also been evidenced in the problem of classification of multiple observation sets, where similar groups of observations are evidenced by the network connections between them. Moreover, the network representation has provided good results in supervised dimensionality reduction in which the intra-class connection matrix and the inter-class penalty matrix have been defined in terms of networked data.

In the semi-supervised paradigm, two techniques have been investigated. The first

one, based on the heuristic of ease of access and on random walk process, has been derived as an extension of the supervised version. In the semi-supervised case, the technique performs transductive classification by propagating labels from a few labeled nodes to unlabeled ones through the underlying network. The second technique, based on interacting forces, can be understood as a dynamical network in which nodes move respecting interaction forces among them. The link strength or force amplitude are calculated in terms of node distance. The network feature evinced in both techniques is the possibility of spreading labels through data disregarding the class shapes, size or densities. The balancing of class sizes and the positioning of labeled data are still open issues, but the network representation can alleviate those challenging problems.

The last investigation reported in this thesis has been on a technique belonging to the unsupervised learning paradigm. As a network-based technique, the data clusters can be detected by the inspection of network subgroups of densely connected nodes. The technique makes usage of consensus time among nodes, which is guided by the network link structure and link weights. In this case, the network helped to derive indirect similarity measures among data.

The reported results have made contributions to the three areas of machine learning. Each of them is specifically commented in the following subsection.

6.1 Conclusions and future studies

In this section, we list the conclusions and future studies for the results reported in the previous chapters.

Conclusions and future works of the network-based supervised learning studies

1. Development a new supervised classification technique which takes into account the ease of access of unlabeled instances to training classes through an underlying network:
 - traditional data classification techniques divide the data space into subspaces (classes) according to physical features of the training data. This work has presented an effort towards a new classification heuristic based on ease of access. Instead of dividing the data attribute space within the training classes, the proposed technique classifies an unlabeled instance with the most easily reached label through an underlying network. To accomplish that, the network-based scheme applies limiting probabilities from random walk theory, which serves as a measure of access to training classes after the

insertion of the unlabeled instance bias into the network link weights. Simulations suggest that the proposed technique is competitive with some well-known and traditional classification techniques. We expect this research can contribute to the network-based learning area, specially to the development of new supervised classification heuristics;

- as future works, special attention should be given to describe and shed more light to the proposed technique behavior, specially to the bias weight composition. This first study considered a linear weight composition between the connection matrices, but a nonlinear combination can be studied to take into account the link densities for example.
2. Development a new supervised classification technique which takes into account the ease of access of unlabeled instances to training classes and the network topological structure to characterize data classes:
 - by using the new classification heuristic based on ease of access and considering also the structural pattern formation of each data class, this work has presented a hybrid network-based classification technique. The proposed method is quite general in the sense that different classification criteria can be considered when constructing the corresponding connection weight matrix of the underlying network. Specifically, we have showed a concrete formulation of the connection weight matrix by combining the physical features (data attributes) and the structural information (network measure) of each class in the training data. We can state that the mixed connection weight matrix provides a comprehensive view of the input data. Furthermore, increasing in accuracy rates has been verified when such a combination is applied. Simulations have suggested that the proposed technique is competitive with some well-known classifiers;
 - in future studies, other measures to describe network structures can be analyzed as well as a mixture among complementary measures. Different network formation techniques can also be explored.
 3. Development of network-based techniques applied to the classification of multiple observation sets:
 - this work has presented two new network-based techniques for the classification of multiple observation sets. In this context, each set of multiple observations corresponds to a single pattern or class. To classify an unlabeled set, the techniques first perform two initial stages: network construction and measure calculation (modularity or Katz index). The network construction

provides a topological representation of the relations among the different patterns. The modularity measure numerically represents the connectedness of the constructed network: the more the network was close to a single densely connected component, the more likely the objects could belong to the same pattern. The Katz centrality measure numerically represents the centrality of the unlabeled observations according to some data class: the more the unlabeled vertices were centrally arranged into a given network, the better the observations, represented by these vertices, were related to that network class. The simulation results have showed that the proposed techniques perform well in handwritten digits and multiple object views collections, overcoming many recent and state-of-the-art multi-view classification methods;

- as future extensions, we suggest the study of different network measures, which can take into account different characteristics of the topological networked representations.
4. Application of different network formation methods into a graph embedding framework to perform supervised dimensionality reduction:
 - we have used a modified version of the recently proposed KAOG network formation method to perform supervised dimensionality reduction. The proposed technique calculates two connection matrices which represent the information of input data about both intra-class and inter-class connections. Both matrices are used into a graph embedding framework which is optimized in terms of a projection vector. Experimental studies have showed that the proposed technique achieves competitive results compared to some other classical network formation methods when applied to dimensionality reduction. It has been shown that the technique enhances data processing by reducing the feature dimension and also increases the classification accuracy;
 - as future studies, more sophisticated methods can be explored in order to create the penalty connection matrix.

Conclusions and future works of the network-based semi-supervised learning studies

1. Extension of the ease of access heuristic to the semi-supervised setting:
 - this work has presented a new network-based semi-supervised classification technique. The set of unlabeled instances compose a network in which

vertices represent the state space for a random walker. Via a specific matrix composition, each labeled instance is inserted into the network to provide a bias to the classification process. This label bias is propagated through the unlabeled vertices by means of the limiting probabilities. The local and global topology are taken into account by the random walk process and thus both clustering and smoothness assumptions are satisfied, making the usage of the unlabeled instances information effective. Simulations have showed that the proposed technique is capable of detecting classes that present different shapes and distributions. The technique has also been demonstrated to be competitive with some well-known semi-supervised techniques using benchmark data sets;

- similarly to the supervised approach, as future works for the semi-supervised extension, special attention should be given to describe and shed more light to the proposed technique behavior, specially to the bias weight composition.

2. Development of a nature-inspired semi-supervised classification technique based on attraction forces:

- this work has presented a new semi-supervised learning technique based on attraction forces among data instances. Labeled instances are considered as fixed attraction points that apply attraction forces on unlabeled instances. In turn, the unlabeled instances are expected to move towards the resultant force direction and, eventually, to converge to an attraction point. Once close enough, the label from the attraction point propagates to the unlabeled neighbor, which becomes a new fixed attraction point. It has been verified that the technique is highly sensitive to the radius of the labeling region and to the dimension of the data set feature space. However, the proposed technique has achieved good classification results in comparison to some well-established classification techniques, even when the semi-supervised smoothness and cluster assumptions were not completely satisfied;
- as future studies, it will be interesting to analyze the behavior of the system when different distance functions are taken into account. Also, different heuristics and refinements for the parameters adjustment, and mechanisms to reduce incorrect labeling during the classification process can be explored.

Conclusions and future works of the network-based unsupervised learning studies

1. Development of a data clustering technique based on synchronization and pinning control of networked dynamical oscillators:
 - this work has presented a new network-based clustering technique consisting of 2 steps: network construction and cluster detection. Networks are constructed by an algorithm which results in connected and sparse networks. Both characteristics improve clustering accuracy by evidencing subgroups in the networks. For cluster detection, the dissimilarities among all vertices are computed via the consensus time dissimilarity, that is, how long it takes vertices to reach a consensus in the presence of a pinned vertex. Finally, the network clusters, that correspond to data clusters, are identified by using the computed dissimilarities. A rigorous mathematical analysis was carried out to provide sufficient conditions on the convergence of the consensus dynamic system. The simulations have showed that the proposed method performs well in the presence of clusters with different sizes and shapes, and is competitive with some classical clustering methods and methods based on spectral analysis. The usage of consensus in networks is a new approach in the field of network-based data clustering and it can be widely explored;
 - as future studies we suggest the analyses of pinning several vertices at the same time, as well as their optimal distribution through the network vertices. The behavior of nonlinear systems representing the vertex dynamics can also be explored.

6.2 List of publications

The investigations reported in this thesis have been published in form of scientific articles in many international journals and conferences. A grouped list by subject showing the related thesis section is given below:

Articles for the supervised learning techniques in Chapter 3

- Articles in international journals:
 1. Cupertino, T. H. and Zhao, L. A unified scheme for data classification using random walk in networks. IEEE Transactions on Neural Networks and Learning Systems. Revised version submitted, 2013.

2. Cupertino, T. H. and Zhao, L. Network-based supervised data classification by using an heuristic of ease of access. *Neurocomputing*. Accepted for publication, 2014.
 3. Cupertino, T. H.; Silva, T. C.; Zhao, L. Classification of multiple observation sets via network modularity. *Neural Computing & Applications (Internet)*, p. 1-7, 2012. DOI: 10.1007/s00521-012-1115-y.
- Publications in international conference proceedings:
 1. Cupertino, T. H.; Carneiro, M. G.; Zhao, L. Dimensionality reduction with the k-associated optimal graph applied to image classification. *IEEE International Conference on Imaging Systems and Techniques (IST)*, October 2013, Beijing, China.
 2. Cupertino, T. H.; Zhao, L. Bias-guided random walk for network-based data classification. In: *Tenth International Symposium on Neural Networks (ISNN)*, 2013, Dalian, China, v. 7952, p. 375-384. DOI: 10.1007/978-3-642-39068-5_46.
 3. Cupertino, T. H.; Zhao, L. Using Katz centrality to classify multiple pattern transformations. In: *2012 Brazilian Symposium on Neural Networks, Curitiba - PR*, v. 1, p. 232-237, DOI: 10.1109/SBRN.2012.23.
 4. Cupertino, T. H.; Silva, T. C.; Zhao, L. Multiple images set classification via network modularity. In: *SIBGRAPI 2012 - Conference on Graphics, Patterns and Images - Workshop of Theses and Dissertations, Ouro Preto - MG*, v. 1, p. 124-129.
 5. Silva, T. C.; Cupertino, T. H. High level classification for pattern recognition. In: *XXIV Sibgrapi Conference on Graphics, Patterns and Images, 2011, Maceió - AL*. Los Alamitos: IEEE Computer Society, 2011. v. 1. p. 344-351. DOI: 10.1109/SIBGRAPI.2011.19.

Articles for the semi-supervised learning techniques in Chapter 4

- Articles in international journals:
 1. Cupertino, T. H.; Gueleri, R.; Zhao, L. A semi-supervised classification technique based on interacting forces. *Neurocomputing (Amsterdam)*, p. 1-9, 2013. DOI: 10.1016/j.neucom.2013.05.050.
- Publications in international conference proceedings:
 1. Cupertino, T. H.; Zhao, L. Semi-supervised learning using random walk limiting probabilities. In: *Tenth International Symposium on Neural Networks*

(ISNN), 2013, Dalian, China, v. 7952, p. 395-404. DOI: 10.1007/978-3-642-39068-5_48.

2. Cupertino, T. H.; Zhao, L. Using interacting forces to perform semi-supervised learning. In: 2012 Brazilian Symposium on Neural Networks, Curitiba - PR, v. 1, p. 91-96, DOI: 10.1109/SBRN.2012.24.

Articles for the unsupervised learning techniques in Chapter 5

- Articles in international journals:

1. Cupertino, T. H.; Huertas, J., Zhao, L. Data clustering using controlled consensus in complex networks. *Neurocomputing (Amsterdam)*, v. 118, p. 132-140, 2013. DOI: 10.1016/j.neucom.2013.02.026.
2. Silva, T. C.; Zhao, L.; Cupertino, T. H. Handwritten data clustering using agents competition in networks. *Journal of Mathematical Imaging and Vision*, v. 45, n. 3, p. 264-276, 2012. DOI: 10.1007/s10851-012-0353-z.

- Publications in international conference proceedings:

1. Silva, T. C.; Cupertino, T. H. Stochastic competitive learning applied to handwritten digit and letter clustering. In: XXIV Sibgrapi Conference on Graphics, Patterns and Images, 2011, Maceió - AL. Los Alamitos: IEEE Computer Society, 2011. v. 1. p. 313-320. DOI: 10.1109/SIBGRAPI.2011.35.

Furthermore, the publications listed below, which have been made during undergraduate scientific research, also contributed to the developments of this thesis:

1. Estepa, R.; Estepa, A.; Cupertino, T. H. A productivity-oriented methodology for local area network design in industrial environments. *Computer Networks (1999)*, v. 55, p. 2303-2314, 2011. DOI: 10.1016/j.comnet.2011.03.011.
2. Estepa, R.; Estepa, A.; Cupertino, T. H.; Vozmediano, J. M.; Madinabeitia, G. A productivity-based approach to LAN topology design. *IEEE Communications Letters (Print)*, v. 15, p. 349-351, 2011. DOI: 10.1109/LCOMM.2011.012511.101742.
3. Cupertino, T. H.; Zhao, L. Traffic congestion on clustered random complex networks. In: 2nd Workshop on Complex Networks, 2010, Rio de Janeiro - RJ. *Communications in Computer and Information Science*. Berlin: Springer, v. 116. p. 13-21. DOI: 10.1007/978-3-642-25501-4_2.
4. Zhao, L.; Cupertino, T. H.; Bertini, J. R. Jr. Chaotic synchronization in general network topology for scene segmentation. *Neurocomputing (Amsterdam)*, v. 71, p. 3360-3366, 2008. DOI: 10.1016/j.neucom.2008.02.024.

5. Cupertino, T. H.; Zhao, L. Minimização de congestionamento de tráfego de pacotes em redes com estruturas complexas. *Revista Eletrônica de Iniciação Científica*. v. 7, n. 2, 2007. ISSN: 1519-8219.
6. Zhao, L.; PARK, K.; Cupertino, T. H.; Lai, Y.-C.; J., Xiaogang. Optimal structure of complex networks for minimizing traffic congestion. *Chaos (Woodbury, N.Y.)*, v. 17, p. 043103-043107, 2007. DOI: 10.1063/1.2790367.
7. Zhao, L.; Park, K.; Lai, Y.-C.; Cupertino, T. H. Attack induced cascading breakdown in complex networks. *Journal of the Brazilian Computer Society (Print)*, v. 13, n. 3, p. 67-76, 2007. DOI: 10.1007/BF03192546.
8. Cupertino, T. H.; Zhao, L. Determining optimal structure of data traffic flow networks. In: *International Joint Conference, I Workshop on Computational Intelligence*, 2006, Ribeirão Preto - SP, v. 1, p. 1-6.

Bibliography

- Afraimovich et al.(1997)** V. S. Afraimovich, S.-N. Chow, and J. K. Hale. Synchronization in lattices of coupled oscillators. *Physica D*, 103:442–451. Cited in page(s) 39
- 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
- Alpaydin(2009)** Ethem Alpaydin. *Introduction to machine learning*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, MA, USA, second edition. Cited in page(s) 2, 20, 22, 23
- Bache and Lichman(2013)** K. Bache and M. Lichman. UCI Machine Learning Repository, 2013. Cited in page(s) 59, 76, 101, 144
- 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, 25
- Barabási(2003)** Albert-László Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume. ISBN 978-0452284395. Cited in page(s) 24
- Barahona and Pecora(2002)** M. Barahona and L. M. Pecora. Synchronization in small-world systems. *Physical Review Letters*, 89:054101. Cited in page(s) 4
- Barrat et al.(2012)** Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. *Dynamical processes on complex networks*. Cambridge University Press, first edition. Cited in page(s) 4, 5
- 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) 116
- 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) 10
- Belykh et al.(2006)** I. Belykh, V. Belykh, and M. Hasler. Synchronization in asymmetrically coupled networks with node balance. *Chaos*, 16:015102(1–9). Cited in page(s) 4, 38

- Bertini et al.(2011)** João Roberto Bertini, Liang Zhao, Robson Motta, and Alneu de Andrade Lopes. A nonparametric classification method based on k -associated graphs. *Information Sciences*, 181:5435–5456. Cited in page(s) 9, 10, 32, 51, 52, 59, 62, 72, 81, 98, 100
- Bhattacharyya(1943)** A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109. Cited in page(s) 86
- Bishop(2007)** Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, second edition. Cited in page(s) 14, 15, 22
- 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) 2, 21
- Boberg and Salakoski(1997)** Jorma Boberg and Tapio Salakoski. Representative noise-free complete-link classification with application to protein structures. *Pattern Recognition*, 30:467–482. Cited in page(s) 144
- Boccaletti et al.(2002)** S. Boccaletti, J. Kurths, G. Osipov, D. L. Valladares, and C. S. Zhou. The synchronization of chaotic systems. *Phys. Rep.*, 366:1–101. Cited in page(s) 4
- Boccaletti et al.(2006)** S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424(4-5):175–308. Cited in page(s) 3, 8
- Bolobás(1985)** B. Bolobás. *Random Graphs*. Academic Press, London. Cited in page(s) 24
- 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) 3
- Breve et al.(2012)** Fabricio Breve, Liang Zhao, Marcos G. Quiles, Witold Pedrycz, and Jiming Liu. Particle competition and cooperation in networks for semi-supervised learning. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9):1686–1698. Cited in page(s) 8
- Brin and Page(1998)** Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117. Cited in page(s) 30
- Bulò et al.(2011)** Samuel Rota Bulò, Massimo Rabbi, and Marcello Pelillo. Content-based image retrieval with relevance feedback using random walks. *Pattern Recognition*, 44(9):2109–2122. Cited in page(s) 41
- Bunke and Riesen(2011)** Horst Bunke and Kaspar Riesen. Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*, 44(9):1928–1940. Cited in page(s) 8
- Burges(1998)** Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167. Cited in page(s) 2, 65

- Burges and Platt(2006)** Christopher J. C. Burges and John C. Platt. *Semi-supervised learning with conditional harmonic mixing*, pages 251–273. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA. Cited in page(s) 116
- Burioni and Cassi(2005)** R. Burioni and D. Cassi. Random walks on graphs: ideas, techniques and results. *J. Phys. A: Math. Gen.*, 38:R45–R78. Cited in page(s) 5
- Çınlar(1975)** Erhan Çınlar. *Introduction to stochastic processes*. Prentice Hall, Englewood Cliffs, NJ, USA. Cited in page(s) 34, 36, 55
- Chapelle and Zien(2005)** O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 57–64. Cited in page(s) 2, 116
- 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) 116
- 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) xxi, 1, 2, 8, 14, 18, 19, 20, 113, 115, 133
- Chehreghani et al.(2008)** Morteza Haghiri Chehreghani, Hassan Abolhassani, and Mostafa Haghiri Chehreghani. Improving density-based methods for hierarchical clustering of web pages. *Data and Knowledge Engineering*, 67:30–50. Cited in page(s) 144
- Chen et al.(2009a)** Fei Chen, Zengqiang Chen, Linying Xiang, Zhongxin Liu, and Zhuzhi Yuan. Reaching a consensus via pinning control. *Automatica*, 45(5):1215–1220. Cited in page(s) 39, 72
- Chen et al.(2009b)** 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) 59, 112
- Chiang et al.(2011)** Ming-Chao Chiang, Chun-Wei Tsai, and Chu-Sing Yang. A time-efficient pattern reduction algorithm for k-means clustering. *Information Sciences*, 181:716–731. Cited in page(s) 146
- Cohen and Havlin(2010)** Reuven Cohen and Shlomo Havlin. *Complex Networks: Structure, Robustness and Function*. Cambridge University Press, Cambridge. Cited in page(s) 3, 10, 24
- Colizza and Vespignani(2008)** V. Colizza and A. Vespignani. Epidemic modeling in metapopulation systems with heterogeneous coupling pattern: theory and simulations. *J. Theor. Biol.*, 251:450–457. Cited in page(s) 4
- Collobert et al.(2006)** Ronan Collobert, Fabian H. Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712. Cited in page(s) 2

- Conaire et al.(2007)** O Conaire, Noel E. O'Connor, and Alan F. Smeaton. An improved spatio-gram similarity measure for robust object localization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 1069–1072. Cited in page(s) 86
- 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) 116
- Cormen et al.(2003)** Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. The MIT Press, Massachusetts, MA, USA. Cited in page(s) 59, 72
- Cortes and Vapnik(1995)** Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, pages 273–297. Cited in page(s) 20
- 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) 17
- Cupertino and Zhao(2012a)** T.H. Cupertino and Liang Zhao. Using interacting forces to perform semi-supervised learning. In *Neural Networks (SBRN), 2012 Brazilian Symposium on*, pages 91–96. Cited in page(s) 115
- Cupertino and Zhao(2012b)** Thiago H. Cupertino and Liang Zhao. Using Katz centrality to classify multiple pattern transformations. In *Neural Networks (SBRN), 2012 Brazilian Symposium on*, pages 232–237. IEEE Computer Society. Cited in page(s) 66, 90, 92
- Cupertino and Zhao(2013a)** Thiago H. Cupertino and Liang Zhao. Bias-guided random walk for network-based data classification. In Chengan Guo, Zeng-Guang Hou, and Zhigang Zeng, editors, *Advances in Neural Networks - ISNN 2013*, volume 7952 of *Lecture Notes in Computer Science*, pages 375–384. Springer Berlin Heidelberg. ISBN 978-3-642-39067-8. Cited in page(s) 55
- Cupertino and Zhao(2013b)** Thiago H. Cupertino and Liang Zhao. Semi-supervised learning using random walk limiting probabilities. In Chengan Guo, Zeng-Guang Hou, and Zhigang Zeng, editors, *Advances in Neural Networks - ISNN 2013*, volume 7952 of *Lecture Notes in Computer Science*, pages 395–404. Springer Berlin Heidelberg. ISBN 978-3-642-39067-8. Cited in page(s) 108
- Cupertino and Zhao(2013c)** Thiago H. Cupertino and Liang Zhao. A unified scheme for data classification using random walk in networks. *Neural Networks and Learning Systems, IEEE Transactions on. Revised version submitted*. Cited in page(s) 66
- Cupertino et al.(2012)** Thiago H. Cupertino, Thiago C. Silva, and Liang Zhao. Classification of multiple observation sets via network modularity. *Neural Computing and Applications (Online)*, 1(1):1–7. Cited in page(s) 66, 90
- Cupertino et al.(2013a)** Thiago H. Cupertino, Murillo G. Carneiro, and Liang Zhao. Dimensionality reduction with the k-Associated Optimal Graph applied to image classification. In *Imaging Systems and Techniques, Proceedings of the IEEE International Conference on*, volume 1, pages 366–371. Cited in page(s) 98

- Cupertino et al.(2013b)** Thiago H. Cupertino, Roberto Guelleri, and Liang Zhao. A semi-supervised classification technique based on interacting forces. *Neurocomputing*, 127(1):43–51. Cited in page(s) 115
- Cupertino et al.(2013c)** Thiago H. Cupertino, Jean Huertas, and Liang Zhao. Data clustering using controlled consensus in complex networks. *Neurocomputing*, 118(1):132–140. Cited in page(s) 135
- Cupertino and Zhao(2011)** Thiago Henrique Cupertino and Liang Zhao. Traffic congestion on clustered random complex networks. In Luciano F. Costa, Alexandre Evsukoff, Giuseppe Mangioni, and Ronaldo Menezes, editors, *Complex Networks*, volume 116 of *Communications in Computer and Information Science*, pages 13–21. Springer Berlin Heidelberg. ISBN 978-3-642-25500-7. Cited in page(s) 5
- 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 2002 International Joint Conference on Neural Networks*, volume 3, pages 2237–2242. IEEE. Cited in page(s) 20
- Dasgupta and Desai(2013)** Bhaskar Dasgupta and Devendra Desai. On the complexity of Newman’s community finding approach for biological and social networks. *J. Comput. Syst. Sci.*, 79(1):50–67. Cited in page(s) 92
- Davidson and Basu(2007)** Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from Data*, 1:1–41. Cited in page(s) 22
- 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) 116
- 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) 20
- 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) 3
- Demšar(2006)** Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30. Cited in page(s) 62, 65, 83
- Diestel(2006)** Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer. Cited in page(s) 25
- Dirickx(2004)** M. Dirickx, editor. *Coherence in complex networks of oscillators*. Cited in page(s) 4
- Djouadi et al.(1990)** Abdelhamid Djouadi, Ö. Snorrason, and F. D. Garber. The quality of training sample estimates of the Bhattacharyya coefficient. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):92–97. Cited in page(s) 86

- Duda et al.(2000)** Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Wiley-Interscience, New York, NY, USA, second edition. Cited in page(s) 2, 14, 20, 23, 65, 98
- 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) 2
- Duin(2000)** R. P. W. Duin. PRTools - Version 3.0 - A Matlab Toolbox for Pattern Recognition. *Proceedings of SPIE*, page 1331. Cited in page(s) 113
- Dunn(1961)** O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64. Cited in page(s) 65, 86
- Erdős and Rényi(1961)** P. Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Math. Acad. Sci. Hungar*, 12:261–267. Cited in page(s) 3
- Fortunato(2010)** Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174. Cited in page(s) 41
- Foster et al.(2001)** Kurt Foster, Stephen Muth, John Potterat, and Richard Rothenberg. A faster Katz status score algorithm. *Computational and Mathematical Organization Theory*, 7:275–285. Cited in page(s) 93
- Freeman(1977)** Linton C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41. Cited in page(s) 9
- Friedman et al.(1997)** N. Friedman, D Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163. Cited in page(s) 2, 65
- Fujino et al.(2005)** Akinori Fujino, Naonori Ueda, and Kazumi Saito. A hybrid generative/discriminative approach to semi-supervised classifier design. In *National Conference on Artificial Intelligence*, pages 764–769. Cited in page(s) 2
- Fujisaka and Yamada(1983)** H. Fujisaka and T. Yamada. Stability theory of synchronized motion in coupled-oscillator systems. *Progress of theoretical physics*, 60:32–47. Cited in page(s) 38
- Fukui and Yamaguchi(2005)** Kazuhiro Fukui and Osamu Yamaguchi. Face recognition using multi-viewpoint patterns for robot vision. In Paolo Dario and Raja Chatila, editors, *Robotics Research*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 192–201. Springer Berlin / Heidelberg. Cited in page(s) 97
- Fukunaga(1991)** Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition edition. Cited in page(s) 18, 98
- Gallager(1995)** Robert G. Gallager. *Discrete stochastic processes*. The Springer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston, MA, USA, first edition. Cited in page(s) 32, 55
- Gameiro and Rodrigues(2001)** M. F. Gameiro and H. M. Rodrigues. Applications of robust synchronization to communication systems. *Applicable Analysis*, 79:21–45. Cited in page(s) 38

- Gan(2007)** Guojun Gan. *Data clustering: theory, algorithms, and applications*, volume 20. Society for Industrial and Applied Mathematics. Cited in page(s) 22
- Gärtner(2008)** Thomas Gärtner. *Kernels for Structured Data*, volume 72. World Scientific Publishing Co., first edition. Cited in page(s) 20
- Gazi and Passino(2003)** Veysel Gazi and Kevin M. Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48:692–697. Cited in page(s) 116
- Grady(2006)** Leo Grady. Random walks for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1768–1783. Cited in page(s) 41
- Grother et al.(1997)** Patrick J. Grother, Gerald T. Candela, and James L. Blue. Fast implementations of nearest neighbor classifiers. *Pattern Recognition*, 30(3):459–465. Cited in page(s) 59, 72
- Guha et al.(1998)** Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84, New York, NY, USA. ACM. ISBN 0-89791-995-5. Cited in page(s) 3
- Guha et al.(2000)** Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *In Proc.ofthe15thInt.Conf.onDataEngineering*. Cited in page(s) 3
- Guimerà et al.(2002)** R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, and A. Arenas. Optimal network topologies for local search with congestion. *Phys. Rev. Lett.*, 89:248701. Cited in page(s) 5
- Hangos et al.(2004)** Katalin M. Hangos, József Bokor, and Gábor Szederkényi. *Analysis and Control of Nonlinear Process Systems*. Springer, 1 edition. Cited in page(s) 120
- Hastie et al.(2009)** Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer Series in Statistics. Springer-Verlag, Berlin, Germany, second edition. Cited in page(s) 18, 62, 81, 83
- Haykin(1998)** Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall, Englewood Cliffs, NJ, USA, second edition. Cited in page(s) 2, 65
- Haykin(2008)** Simon S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition. Cited in page(s) 18
- Hsu and Lin(2002)** Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *Neural Networks, IEEE Transactions on*, 13:415–425. Cited in page(s) 62, 83
- Hubert and Arabie(1985)** Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218. Cited in page(s) 144
- Jain(2010)** Anil K. Jain. Data Clustering: 50 Years Beyond K-Means. *Pattern Recognition Letters*, 31:651–666. Cited in page(s) 2
- Jain and Dubes(1988)** Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, New York, NY, USA. Cited in page(s) 144

- Jain et al.(1999)** Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323. Cited in page(s) 23
- Joachims(2003)** Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 290–297. AAAI Press. Cited in page(s) 116
- Jolliffe(2002)** Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, second edition. Cited in page(s) 23, 97
- Karypis et al.(1999)** George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75. Cited in page(s) 8
- Katz(1953)** Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43. Cited in page(s) 10, 30
- Kim(2009)** Ji-Hyun Kim. Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics and Data Analysis*, 53:3735–3745. Cited in page(s) 60, 62, 77, 81, 102
- 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) 37
- Kokiopoulou and Frossard(2010)** Effrosini Kokiopoulou and Pascal Frossard. Graph-based classification of multiple observation sets. *Pattern Recognition*, 43:3988–3997. Cited in page(s) 49, 50, 90, 93, 94, 96, 97
- Koller and Friedman(2009)** Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, USA, 1st edition. Cited in page(s) 17
- Kryszkiewicz and Skonieczny(2005)** Marzena Kryszkiewicz and Lukasz Skonieczny. Faster clustering with DBSCAN. *Intelligent Information Processing and Web Mining*, pages 605–614. Cited in page(s) 3
- LaSalle(1986)** Joseph P. LaSalle. *The Stability and Control of Discrete Processes*, volume 62 of *Applied Mathematical Sciences*. Springer-Verlag, New York, NY. Cited in page(s) 139
- LeCun et al.(1998)** Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324. Cited in page(s) 59, 76
- Leibe and Schiele(2003)** Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–409–15. Cited in page(s) 86, 96
- Li et al.(2008)** Kezan Li, Michael Small, and Xinchu Fu. Generation of clusters in complex dynamical networks via pinning control. *Journal of Physics A: Mathematical and Theoretical*, 41(50):505101. Cited in page(s) 9

- Li and Chen(2003)** Xiang Li and Guanrong Chen. Synchronization and desynchronization of complex dynamical networks: an engineering viewpoint. *Circuits and Systems I - Regular Papers, IEEE Transactions on*, 50:1381–1390. Cited in page(s) 137
- Li and Wang(2004)** Xiang Li and Xiaofan Wang. Feedback control of scale-free coupled Henon maps. In *International Conference on Control, Automation, Robotics and Vision*, pages 574–578. Cited in page(s) 137
- 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) 37
- Linde and Buzo(1980)** Y. Linde and A. Buzo. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28:84–95. Cited in page(s) 146
- Liu et al.(2003)** Yang Liu, K. M. Passino, and M. Polycarpou. Stability analysis of one-dimensional asynchronous swarms. *IEEE Transactions on Automatic Control*, 48:1848–1854. Cited in page(s) 116
- Loeff et al.(2008)** Nicolas Loeff, David A. Forsyth, and Deepak Ramachandran. Manifoldboost: stagewise function approximation for fully-, semi- and un-supervised learning. In *International Conference on Machine Learning*, pages 600–607. Cited in page(s) 2
- Lovász et al.(2003)** L. Lovász, J. Pelikán, and K. Vesztergombi. *Discrete Mathematics*. Springer. ISBN 978-85-85818-28-X. Cited in page(s) 24, 25
- Luxburg(2007)** Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416. Cited in page(s) 146
- 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) 2
- Mallapragada et al.(2009)** Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2000–2014. Cited in page(s) 2
- Miller and Uyar(1996)** David J. Miller and Hasan S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Neural Information Processing Systems*, pages 571–577. Cited in page(s) 2
- Mitchell(1997)** Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, first edition. Cited in page(s) 1, 2, 14, 15, 21, 22, 23
- Mitchell(1999)** Tom M. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*. Cited in page(s) 21
- Moreno and Pacheco(2004)** Nekovee M. Moreno, Y. and A. F. Pacheco. Dynamics of rumor spreading in complex networks. *Physical Review E*, 69:066130. Cited in page(s) 4

- Motter et al.(2002)** A. E. Motter, C. Zhou, and Kurths. Network synchronization, diffusion, and the paradox of heterogeneity. *Physical Review E*, 71:016116. Cited in page(s) 4
- Motter and Lai(2002)** Adilson E. Motter and Ying-Cheng Lai. Cascade-based attacks on complex networks. *Phys. Rev. E*, 66:065102. Cited in page(s) 5
- Neapolitan(2003)** Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, first edition. Cited in page(s) 17
- Newman(2001)** M. E. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Phys Rev E Stat Nonlin Soft Matter Phys*, 64(1 Pt 2). Cited in page(s) 29
- Newman(2003a)** Mark E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126. Cited in page(s) 3, 28
- Newman(2004)** Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133. Cited in page(s) 9
- Newman(2006)** 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) 9
- Newman(2003b)** Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256. Cited in page(s) 3
- Newman and Girvan(2004)** Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113(1–15). Cited in page(s) 9, 28, 66
- Ng et al.(2001)** Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press. Cited in page(s) 146
- Ng and Han(2002)** Raymond T. Ng and Jiawei Han. CLARANS: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016. Cited in page(s) 2
- Nguyen and Mamitsuka(2011)** Canh Hao Nguyen and H. Mamitsuka. Discriminative graph embedding for label propagation. *Neural Networks, IEEE Transactions on*, 22(9):1395–1405. Cited in page(s) 8
- Ni et al.(2012)** Bingbing Ni, Shuicheng Yan, and A. Kassim. Learning a propagable graph for semisupervised learning: classification and regression. *Knowledge and Data Engineering, IEEE Transactions on*, 24(1):114–126. Cited in page(s) 8
- Nigam et al.(2000a)** Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, pages 1–34. Cited in page(s) 2
- Nigam et al.(2000b)** Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2-3):103–134. Cited in page(s) 20

- Noh and Rieger(2004)** J. D. Noh and H. Rieger. Random walks on complex networks. *Physical Review Letters*, 92:118701. Cited in page(s) 5
- Olfati-Saber and Murray(2004)** R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533. Cited in page(s) 39
- Olfati-Saber et al.(2007)** R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233. Cited in page(s) 39
- Oliveira and Zhao(2008)** Tatyana B. S. Oliveira and Liang Zhao. Complex network community detection based on swarm aggregation. *Proceedings of the Fourth International Conference on Natural Computation*, 7:604–608. Cited in page(s) 9
- 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*, 435:814–818. Cited in page(s) 8
- Pecora and Carroll(1990)** L. M. Pecora and T. L. Carroll. Synchronization in chaotic systems. *Phys. Rev. Lett.*, 64:821–824. Cited in page(s) 4, 38
- Pernkopf et al.(2012)** F. Pernkopf, M. Wohlmayr, and S. Tschitschek. Maximum margin bayesian network classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):521–532. Cited in page(s) 17
- Phillip and Bonacich(2007)** Phillip and Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555 – 564. Cited in page(s) 31
- Pikovsky and Kurths(2001)** Rosenblum M. Pikovsky, A. and J. Kurths. *Synchronization - a universal concept in nonlinear sciences*. Cambridge Univ. Press. Cited in page(s) 4, 38
- Platt(1999)** J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, Cambridge, MA, USA. MIT Press. Cited in page(s) 59, 72
- 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, 41, 43
- Quinlan(1986)** J. R. Quinlan. Induction of decision trees. *Machine learning*, 1:81–106. Cited in page(s) 17
- Quinlan(1987)** J. Ross Quinlan. Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 304–307. Cited in page(s) 17
- Quinlan(1992)** John R. Quinlan. *C4.5: programs for machine learning*. Series in Machine Learning. Morgan Kaufman Publishers, first edition. Cited in page(s) 17, 59, 62, 65, 72, 81
- Rand(1971)** William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850. Cited in page(s) 144

- Riesen and Bunke(2009)** Kaspar Riesen and Horst Bunke. Graph Classification by Means of Lipschitz Embedding. *IEEE Transactions on Systems, Man, and Cybernetics*, 39:1472–1483. Cited in page(s) 8
- Rodrigues et al.(2001)** H. M. Rodrigues, F. C. L. Alberto, and N. G. Bretas. Uniform invariance principle and synchronization. robustness with respect to parameter variation. *J. Diff. Eq.*, 169:228–254. Cited in page(s) 38
- Roos et al.(2005)** Teemu Roos, Hannes Wettig, Peter Grünwald, Petri Myllymäki, and Henry Tirri. On discriminative bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296. Cited in page(s) 17
- Rosenblatt(1957)** Frank Rosenblatt. The perceptron: a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, NY. Cited in page(s) 18
- Rulkov et al.(1995)** N. F. Rulkov, M. M. Sushchik, L. S. Tsimring, and H. D. I. Abarbanel. Generalized synchronization of chaos in directionally coupled chaotic systems. *Physical Review E*, 51:980–994. Cited in page(s) 4
- Sakano and Mukawa(2000)** Hitoshi Sakano and Nmki. Mukawa. Kernel mutual subspace method for robust facial image recognition. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 245–248. Cited in page(s) 97
- Schaeffer(2007)** Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1): 27–64. Cited in page(s) 8
- Seeger(2002)** M. Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh. Cited in page(s) 15
- Shahshahani and Landgrebe(1994)** B.M. Shahshahani and D.A. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *Geoscience and Remote Sensing, IEEE Transactions on*, 32(5):1087–1095. Cited in page(s) 2
- Shakhnarovich et al.(2002)** Gregory Shakhnarovich, John Fisher, and Trevor Darrell. Face recognition from long-term observations. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision — ECCV 2002*, volume 2352 of *Lecture Notes in Computer Science*, pages 851–865. Springer Berlin / Heidelberg. Cited in page(s) 97
- Shi and Malik(2000)** Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905. Cited in page(s) 146
- Sibson(1973)** R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16:30–34. Cited in page(s) 141
- Silva and Zhao(2012a)** Thiago C. Silva and Liang Zhao. Network-based stochastic semisupervised learning. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(3):451–466. Cited in page(s) 8, 41, 43

- Silva and Zhao(2012b)** Thiago C. Silva and Liang Zhao. Stochastic competitive learning in complex networks. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(3):385–398. Cited in page(s) 8, 41, 43, 44
- Silva and Zhao(2012c)** Thiago C. Silva and Liang Zhao. Network-based high level data classification. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(6): 954–970. Cited in page(s) 66
- Silva et al.(2013)** Thiago C. Silva, Liang Zhao, and Thiago H. Cupertino. Handwritten data clustering using agents competition in networks. *Journal of Mathematical Imaging and Vision*, 45(3):264–276. Cited in page(s) 8, 9, 41, 43
- 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) 10, 116
- Spirin and Mirny(2003)** V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences (USA)*, 100(21):12123–12128. Cited in page(s) 8
- 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) 37
- Su and Zhang(2006)** J. Su and H. Zhang. A fast decision tree learning algorithm. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1, pages 500–505. Cited in page(s) 59, 72
- 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(4):681–699. Cited in page(s) 116
- Tan et al.(2005)** Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Addison-Wesley, Boston, MA, USA, first edition. Cited in page(s) 2, 60, 65, 81
- Theodoridis and Koutroumbas(2008)** Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition. Cited in page(s) 1
- Timme et al.(2002)** M. Timme, F. Wolf, and T. Geisel. Coexistence of regular and irregular dynamics in complex networks of pulse-coupled oscillators. *Physical Review Letters*, 89:258701. Cited in page(s) 4
- Tsang et al.(2005)** I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392. Cited in page(s) 59, 72
- Vapnik(1999)** Vladimir Vapnik. *The nature of statistical learning theory*. Springer-Verlag, Berlin, Germany, second edition. Cited in page(s) 62, 81
- Vapnik(1998)** Vladimir N. Vapnik. *Statistical learning theory*. Wiley-Interscience, New York, first edition. Cited in page(s) 18, 20

- Vinh et al.(2009)** Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 1073–1080, New York, NY, USA. ACM. ISBN 978-1-60558-516-1. Cited in page(s) 144
- Vogels and Birman(2003)** van Renesse R. Vogels, W. and K. Birman. The power of epidemics: robust communication for large-scale distributed systems. *SIGCOMM Comput. Commun.*, 33(1):131–135. Cited in page(s) 4
- Wang and Zhang(2008)** Fei Wang and Changshui Zhang. Label Propagation through Linear Neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1): 55–67. Cited in page(s) 115, 116
- 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) 25
- 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, 25, 70
- Wechsler and Citron(1980)** Harry Wechsler and Todd Citron. Feature extraction for texture classification. *Pattern Recognition*, 12(5):301–311. Cited in page(s) 41
- Wechsler and Kidode(1979)** Harry Wechsler and Masatsugu Kidode. A random walk procedure for texture discrimination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1(3):272–280. Cited in page(s) 41
- Xiang et al.(2007)** L.Y. Xiang, Z.X. Liu, Z.Q. Chen, F. Chen, and Z.Z. Yuan. Pinning control of complex dynamical networks with general topology. *Physica A: Statistical Mechanics and its Applications*, 379(1):298–306. Cited in page(s) 40
- Yan et al.(2007)** Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):40–51. Cited in page(s) 10, 98, 99
- Zanette(2002)** D. H. Zanette. Dynamics of rumor propagation on small-world networks. *Physical Review E*, 65:041908. Cited in page(s) 4
- Zhan(2005)** Xingzhi Zhan. Extremal Eigenvalues of Real Symmetric Matrices with Entries in an Interval. *Siam Journal on Matrix Analysis and Applications*, 27:851–860. Cited in page(s) 139
- Zhang et al.(2010)** Haifeng Zhang, Kezan Li, and Xinchu Fu. On pinning control of some typical discrete-time dynamical networks. *Communications in Nonlinear Science and Numerical Simulation*, 15:182–188. Cited in page(s) 137
- 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) 5, 29

- Zhao et al.(2005a)** Liang Zhao, Ying-Cheng Lai, Kwangho Park, and Nong Ye. Onset of traffic congestion in complex networks. *Phys. Rev. E*, 71:026125. Cited in page(s) 5, 29
- Zhao et al.(2005b)** 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) 5, 29
- 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) 5, 29
- Zhao et al.(2008a)** Liang Zhao, Thiago H. Cupertino, and Jo ao R. Bertini Jr. Chaotic synchronization in general network topology for scene segmentation. *Neurocomputing*, 71(16-18):3360–3366. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006). Cited in page(s) 9, 38, 45, 46
- Zhao et al.(2008b)** Tao Zhao, Ramakant Nevatia, and Bo Wu. Segmentation and tracking of multiple humans in crowded environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1198–1211. Cited in page(s) 10
- Zhou and Schölkopf(2006)** D. Zhou and B. Schölkopf. Discrete Regularization. In Olivie Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-supervised Learning*, pages 237–250. MIT Press, Cambridge, MA, USA. Cited in page(s) 116
- Zhou and Schölkopf(2004)** Dengyong Zhou and Bernhard Schölkopf. Learning from labeled and unlabeled data using random walks. In *Pattern Recognition, Proceedings of the 26th DAGM Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 237–244. Springer Berlin / Heidelberg. Cited in page(s) 115, 116
- Zhou et al.(2003)** 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 16*, volume 16, pages 321–328. MIT Press. Cited in page(s) 47, 48, 49, 94
- Zhou(2003a)** Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(6):061901. Cited in page(s) 9
- Zhou(2003b)** Haijun Zhou. Network landscape from a Brownian particle’s perspective. *Physical Review E*, 67(4):041908. Cited in page(s) 9
- Zhu(2005)** Xiaojin Zhu. Semi-supervised learning with graphs. *Doctoral Thesis - Carnegie Mellon University. CMU-LTI-05-192*. Cited in page(s) 31
- Zhu(2008)** Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin, Madison. Cited in page(s) 20, 21
- 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) 116
- 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) 15, 20