UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Collective dynamics in complex networks for machine learning

Filipe Alves Neto Verri

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Filipe Alves Neto Verri

Collective dynamics in complex networks for machine learning

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Zhao Liang

USP – São Carlos
April 2018

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

V554c

Verri, Filipe Alves Neto
Collective dynamics in complex networks for machine learning / Filipe Alves Neto Verri; orientador Zhao Liang. -- São Carlos, 2018.
95 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, 2018.

1. Aprendizado de máquina. 2. Dinâmica coletiva. 3. Redes complexas. 4. Aprendizado semissupervisionado. 5. Aprendizado não supervisionado. I. Liang, Zhao, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Filipe Alves Neto Verri

Dinâmica coletiva em redes complexas para aprendizado de máquina

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*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Zhao Liang

USP – São Carlos
Abril de 2018

ACKNOWLEDGEMENTS

I would like to thank my wife, Nayara, for her unconditional support since the beginning of this journey.

Also, I would like to express my gratitude towards my friends and colleagues that have spent a lot of time with me “doing science”.

For FAPESP, I sincerely give my thanks for the continuous financial support. (Grants 2013/25876-6 and 2015/18456-6.)

I would like to express my gratitude to the University of São Paulo and the Institute of Mathematics and Computer Sciences for hosting me through my B.Sc. and Ph.D. period.

Of course, I would like to thank deeply my thesis advisor, Zhao Liang, who has provided invaluable pieces of advice and guidance in the project.

Above all, I thank God for reaching me out and guiding my life. I praise Him for the wonderful gifts and miracles He has done.

ABSTRACT

VERRI, F. A. N. **Collective dynamics in complex networks for machine learning**. 2018. 95 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Machine learning enables machines to learn automatically from data. In literature, graph-based methods have received increasing attention due to their ability to learn from both local and global information. In these methods, each data instance is represented by a vertex and is linked to other vertices according to a predefined affinity rule. However, they usually have unfeasible time cost for large problems. To overcome this problem, techniques can employ a heuristic to find suboptimal solutions in a feasible time. Early heuristic optimization methods exploit nature-inspired collective processes, such as ants looking for food sources and swarms of bees. Nowadays, advances in the field of complex systems provide powerful tools to assess and to understand dynamical systems. Complex networks, which are graphs with nontrivial topology, are among these theoretical tools capable of describing the interplay of topology, structure, and dynamics of complex systems. Therefore, machine learning methods based on complex networks and collective dynamics have been proposed. They encompass three steps. First, a complex network is constructed from the input data. Then, the simulation of a distributed collective system in the network generates rich information. Finally, the collected information is used to solve the learning problem. The coordination of the individuals in the system permit to achieve dynamics that is far more complex than the behavior of single individuals. In this research, I have explored collective dynamics in machine learning tasks, both in unsupervised and semi-supervised scenarios. Specifically, I have proposed a new collective system of competing particles that shifts the traditional vertex-centric dynamics to a more informative edge-centric one. Moreover, it is the first particle competition system applied in machine learning task that has deterministic behavior. Results show several advantages of the edge-centric model, including the ability to acquire more information about overlapping areas, a better exploration behavior, and a faster convergence time. Also, I have proposed a new network formation technique that is not based on similarity and has low computational cost. Since addition and removal of samples in the network is cheap, it can be used in real-time application. Finally, I have conducted analytical investigations of a flocking-like system that was needed to guarantee the expected behavior in community detection tasks. In conclusion, the result of the research contributes to many areas of machine learning and complex systems.

Keywords: Machine learning, Collective dynamics, Complex networks, Semi-supervised learning, Unsupervised learning.

RESUMO

VERRI, F. A. N. **Dinâmica coletiva em redes complexas para aprendizado de máquina**. 2018. 95 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Aprendizado de máquina permite que computadores aprendam automaticamente dos dados. Na literatura, métodos baseados em grafos recebem crescente atenção por serem capazes de aprender através de informações locais e globais. Nestes métodos, cada item de dado é um vértice e as conexões são dadas uma regra de afinidade. Todavia, tais técnicas possuem custo de tempo impraticável para grandes grafos. O uso de heurísticas supera este problema, encontrando soluções subótimas em tempo factível. No início, alguns métodos de otimização inspiraram suas heurísticas em processos naturais coletivos, como formigas procurando por comida e enxames de abelhas. Atualmente, os avanços na área de sistemas complexos provêm ferramentas para medir e entender estes sistemas. Redes complexas, as quais são grafos com topologia não trivial, são uma das ferramentas. Elas são capazes de descrever as relações entre topologia, estrutura e dinâmica de sistemas complexos. Deste modo, novos métodos de aprendizado baseados em redes complexas e dinâmica coletiva vêm surgindo. Eles atuam em três passos. Primeiro, uma rede complexa é construída da entrada. Então, simula-se um sistema coletivo distribuído na rede para obter informações. Enfim, a informação coletada é utilizada para resolver o problema. A interação entre indivíduos no sistema permite alcançar uma dinâmica muito mais complexa do que o comportamento individual. Nesta pesquisa, estudei o uso de dinâmica coletiva em problemas de aprendizado de máquina, tanto em casos não supervisionados como semissupervisionados. Especificamente, propus um novo sistema de competição de partículas cuja competição ocorre em arestas ao invés de vértices, aumentando a informação do sistema. Ainda, o sistema proposto é o primeiro modelo de competição de partículas aplicado em aprendizado de máquina com comportamento determinístico. Resultados comprovam várias vantagens do modelo em arestas, includindo detecção de áreas sobrepostas, melhor exploração do espaço e convergência mais rápida. Além disso, apresento uma nova técnica de formação de redes que não é baseada na similaridade dos dados e possui baixa complexidade computational. Uma vez que o custo de inserção e remoção de exemplos na rede é barato, o método pode ser aplicado em aplicações de tempo real. Finalmente, conduzi um estudo analítico em um sistema de alinhamento de partículas. O estudo foi necessário para garantir o comportamento esperado na aplicação do sistema em problemas de detecção de comunidades. Em suma, os resultados da pesquisa contribuíram para várias áreas de aprendizado de máquina e sistemas complexos.

Palavras-chave: Aprendizado de máquina, Dinâmica coletiva, Redes complexas, Aprendizado semissupervisionado, Aprendizado não supervisionado.

LIST OF FIGURES

CONTENTS

CHAPTER

1

INTRODUCTION

Human beings possess a learning capability far beyond other animals. The reasons behind such ability bring several questions to the scientific community. Formally, *learning* is the process of developing new and enhancing existing abilities. A *learning theory* is a conceptual model to describe how people learn (MOWRER, 1960). In Psychology, traditional theories attempt to explain learning as a process of acquiring, evaluating, and applying experience. Although such theories provide a full description of the individual learning process, most of them fail to address the sudden shifts in the environment – which people barely control. Learning is cyclical and continuous. Individuals connect to a network to share and access new information, modifying not only their beliefs but also the network itself; once they share their understandings, they find more new information. Recent theories study these dynamic properties of learning. *Connectivism*, for instance, model the *structure of knowledge* as a network and *learning* as a process of pattern recognition (ALDAHDOUH; OSÓRIO; CAIRES, 2015).

Learning, however, is not restricted to animals. Machines can obtain knowledge from experience too. *Machine learning* is the science that enables machines to learn automatically from data. Learning methods have been successfully applied in astronomy (BANERJI *et al.*, 2010; DEVINE; GOSEVA-POPSTOJANOVA; MCLAUGHLIN, 2016), health care (SAU; BHAKTA, 2017; MERTZ, 2018), driving assistance (MA; XIE; BROWN, 2018), robotics (AGRIOMALLOS *et al.*, 2018), seismology (REYNEN; AUDET, 2017), and many other industry and scientific fields (HAYKIN, 2009). One specific task machines can learn is the assignment of labels to given input values (BISHOP, 2006). Depending on the information provided in the input data, the label-assignment tasks are divided into *data clustering* and *data classification*.

Data classification, or simply classification, is the problem of categorizing inputs using labels that belong to a discrete set. In this scenario, the labels are called *classes*. To perform classification, learning systems count on a training set comprising data whose labels are known totally (supervised) or partially (semi-supervised).

Figure 1 – Different label information in learning problems. Each point corresponds to a data sample in a bidimensional Euclidean space. Samples whose label information is unknown are depicted in gray. Every other color represents a class (label).

Data clustering or clustering is the unsupervised equivalent of classification. That means that no label information is provided for the input data. Labels, in this case, indicate which group or cluster the input belongs to. Since the notion of a cluster is not precisely defined, clustering systems group data according to some similarity definition. For example, data can be grouped based on their distance, distribution, or density.

Figure 1 illustrates the possible scenarios according to the input label information. In the supervised case, the goal is to build a model, called classifier, that output the classes of unseen samples, generalizing the acquired knowledge. In the second case, semi-supervised, the goal is either to build a classifier (induction) or to propagate the labels to the input unlabeled samples (transduction). In the last case, with no label information, input samples are grouped by some similarity rule in an unsupervised manner.

Semi-supervised classification lies in the semi-supervised learning (SSL) paradigm. SSL is an important research field since, in real-world applications, one usually has partial knowledge on a given data set. For example, a credit card provider cannot be sure about the legitimacy of every transaction, but it can assume that some are frauds; in the genetics field, one certainly does not know the functions of all genes, but the functions of some of them are known. Sometimes, although one has a complete or almost complete knowledge of a data set, labeling it by hand is lengthy and expensive. So it is often necessary to restrict the labeling scope. SSL methods are capable of exploiting both unlabeled and labeled data to solve the problem.

Many semi-supervised learning techniques have been developed, including generative models (NIGAM *et al.*, 2000), discriminative models (LOOG; JENSEN, 2015), clustering and labeling techniques (WAGSTAFF *et al.*, 2001), multi-training (ZHOU; LI, 2005), low-density

separation models (VAPNIK, 1998), and graph-based methods (SILVA; ZHAO, 2012c; CHENG; PAN, 2014; ZHANG *et al.*, 2015). In the graph-based methods, each data instance is represented by a vertex and is linked to other vertices according to a predefined affinity rule. The labels are propagated from a few labeled vertices to the whole graph using a particular optimization heuristic (BELKIN; NIYOGI; SINDHWANI, 2006).

Graph-based methods have received increasing attention due to their ability to learn from both local and global information. Many of them estimate a function on the graph that maps each vertex into a label. The estimated function satisfies two objectives at the same time: it is close to the known labels on the labeled nodes, and it should be smooth on the whole graph (ZHU, 2005).

From the point of view of mathematical optimization, those graph-based techniques are regularization frameworks where the first term is a loss function and the second term is a regularizer (ZHU, 2005). Although there are many optimization methods to deal with this problem, exact solvers usually have unfeasible computational cost for large graphs. An alternative is relying on a heuristic to find suboptimal solutions in a feasible time.

Several dynamic processes that exhibit complex and self-organizing behavior exist in nature (JENSEN, 1998). They often minimize a cost inherently, without the need for a global controller. For example, ants minimize the distance between their nest and a food source using a decentralized pheromone strategy (DORIGO; BIRATTARI, 2010); birds organize themselves to reduce air resistance and to save energy with no leader (KENNEDY, 2006); evolution itself is a process that adapts species to their environment without global orchestration (BÄCK, 1996).

An interesting characteristic of such collective systems is that patterns emerge without external control. At the macroscopic level, the group of individuals displays a robust and coherent pattern. However, at the microscopic level, the individuals react stochastically to the partial information provided by neighboring individuals (MOUSSAID *et al.*, 2009). The coordination and interaction of the individuals in the system permit to achieve dynamics that is far more complex than the behavior of single individuals. Such phenomenon is called *emergence* (HOLLAND, 1999).

Early heuristic optimization methods, such as genetic algorithms (GA) (KRAMER, 2017) and ant colony optimization (ACO) (DORIGO; BIRATTARI, 2010), exploit these nature- and bio-inspired processes by simulating a simplified version of them. The simulation is often performed by an agent-based model (ABM), that is, a computational model for the actions and interactions of autonomous agents with the intent to assess their effects on the system as a whole.

Nowadays, advances in the field of complex systems give us powerful tools to assess and to understand dynamical systems better than via ABM simulations. Complex networks are among the theoretical tools that have been receiving a lot of attention in the scientific community. Complex networks are (usually large-scale) graphs with nontrivial topology (NEWMAN; BARABÁSI; WATTS, 2006). They are a powerful tool to describe the interplay of topology,

structure, and dynamics of complex systems (NEWMAN; BARABÁSI; WATTS, 2006; NEWMAN, 2010). Therefore, they provide a groundbreaking strategy to understand the behavior of many real-world systems. Networks are also an important mechanism for data representation and analysis (SILVA; ZHAO, 2016). Interpreting datasets as complex networks enables us to access the interrelational nature of data items further, since several structural and dynamical characterization techniques for complex networks have been studied deeply.

Attaching a nonlinear collective dynamics in complex networks generates rich information. Such a redundant and distributed processing handles the adaptability and robustness of the learning process. The complex-network framework provides the tools to control and to analyze the obtained information. As a result, learning models based on collective systems not only perform as well as (or even better than) graph-based SSL methods, but also overcome their high computational cost. While the computational complexity of graph-based techniques are usually at cubic order (ZHU; GOLDBERG; KHOT, 2009), collective distributed ones are subquadratic (SILVA; ZHAO, 2016). Common strategies to reduce the complexity by sacrificing part of the available information, such as using a set of sparse prototypes derived from the data (ZHANG *et al.*, 2015), are not needed when complex network approaches are used.

The use of collective dynamics is not restricted to SSL problems. Since clustering is a special case of SSL – considering that the number of labeled instances goes to zero – few adaptations are needed in these techniques to deal with unsupervised problems. Since the input data is converted into a complex network, the clustering task is equivalent to the task of community detection in graphs. Formally, communities are groups of densely connected vertices, while connections between different communities are sparser (FORTUNATO, 2010). Finding the optimal partition is an NP-hard problem in most cases (FORTUNATO, 2010), thus making interesting the use of heuristics to find out suboptimal solutions in feasible time. Methods based on collective dynamics excel in such task as well (SILVA; ZHAO, 2016).

Learning techniques that extract information from collective dynamics in complex networks have been applied in image segmentation (ZHAO; MACAU, 2001; ZHAO, 2003; DAMIANCE; ZHAO; CARVALHO, 2004), community detection (QUILES *et al.*, 2008; MORARESCU; GIRARD, 2011; BREVE; ZHAO, 2013) and network characterization (ARENAS; DÍAZ-GUILERA; PÉREZ-VICENTE, 2006), semi-supervised classification (SILVA; ZHAO, 2012b; CUPERTINO; GUELERI; ZHAO, 2014; URIO; VERRI; ZHAO, 2016), data clustering (CUPERTINO; HUERTAS; ZHAO, 2013), and robotics (PFEIFER; LUNGARELLA; IIDA, 2007).

Besides the advances in machine learning based on graphs or complex networks, one subject that has received little attention is the network formation step. All of those methods require that the input dataset is in a graph format. When the input data does not comply with this requirement, an additional network formation step is mandatory; and such step is as important as the learning model itself (ZHU, 2005; SILVA; ZHAO, 2016). A good balance between the (time and memory) computational cost and the amount of information kept is essential.

Most of the strategies transform each input sample into a vertex and connect them based on a similarity (or distance) strategy. The two most used approaches are the k-nearest-neighbors graph, that connects each sample to the k most similar ones, and the ε-radius graph, that connects each sample to all other with distance less than (or equal to) ε (SILVA; ZHAO, 2016). Alternatives include combinations of both methods (SILVA; ZHAO, 2012a), directed and weighted approaches (NETO; ZHAO, 2013), and nontrivial distribution of the edges according to some heuristic (JEBARA; WANG; CHANG, 2009; BERTINI *et al.*, 2011). Given the problem-specific nature of this step, there is much room for exploration.

1.1 Objectives

My goal is to study learning models based on complex networks and collective dynamics. My hypothesis is that the theory of complex systems provides better understanding of these machine learning models. Such understanding enables both the improvement of the existing methods and the development of new methods that overcome common problems in the field.

During my investigations, I tackled three specific topics:

- Improvements in particle competition methods and the study of edge-centric approaches.
- Proposal of a bipartite-network formation method and the adaptation of collective learning methods in such networks.
- Analytical studies on a flocking-like system and its application in community detection problems.

1.2 Motivations

The motivations and hypotheses for the study of the aforementioned topics are discussed in this section.

1.2.1 Particle competition

Competition is a dynamic process observed in many biological and social systems that have limited resources, such as water, food, mates, territory, fame, etc. Competitive learning is an important learning approach that has been applied mostly in the field of neural networks and in unsupervised tasks. Early contributions are self-organizing maps (KOHONEN, 1998), stochastic competitive learning (KOSKO, 1991), and adaptive resonance theory (GROSSBERG, 1987). Recent approaches based on complex networks and collective systems (QUILES *et al.*, 2008; SILVA; ZHAO, 2012b) address two main limitations of the early methods: the lack of robustness

for data processing, caused by the limited size of the network; and the lack of correspondence between the original data and the network, acting as “black box” systems.

The newer methods deal with a wide range of unsupervised and semi-supervised applications (SILVA; ZHAO, 2016). However, models in the literature concern vertex dynamics, that is, how each vertex changes its state. Intuitively, vertex dynamics is a rough modeling of a network because each vertex can have several edges. A problem with the data analysis in this approach is the overlapping nature of the vertices, where a data sample can belong to more than one class. Therefore, it is interesting to know how each edge changes its state in the competition process to acquire detailed knowledge of the dynamical system. My hypothesis is that edge-centric particle competition systems exhibit better behavior for the application in machine learning.

Moreover, particle competition models are inherently stochastic. As a result, learning methods based on them are not deterministic and can output different solutions for the same inputs in different runs. In order to guarantee an (almost-)stable learning, one needs to run the algorithm several times and average the results. Thus, the further analytical study of such systems may provide insights to find out deterministic evolutions with the same desired properties. The premise is that such deterministic version exists and maintains itself useful in machine learning problems.

1.2.2 Network formation

Methods based on complex networks assume the input data is represented by a graph. As in other machine learning techniques, the quality of the input graph contributes greatly to acquire knowledge (ZHU, 2005). Naïve network formation techniques usually depend on a neighborhood parameter and on a distance metric. Choosing a metric is often a simple task that depends on the domain of application. However, it is not trivial to choose the neighborhood parameter. Although some researches propose heuristics to estimate good neighborhoods (TENENBAUM, 2000; JEBARA; WANG; CHANG, 2009), it is hard to achieve a sparse and yet expressive network. Moreover, all existing methods rely on the awareness of some notion of similarity between samples. Calculating such metric may be costly.

Therefore, there is much room for improvement in this step, particularly regarding new methods with low computational cost and with intuitive parameters. We assume that new methods of network formation can be developed without the need of relying on the similarity of the input samples. Also, that the existing dynamical systems can be adapted to retrieve information from the new kinds of networks.

1.2.3 Flocking systems

Flocking and swarm behavior are phenomena present in many ecological collective systems, from bacteria to large animals such as fish, birds, and bats. These systems are usu-

ally modeled by the dynamics of self-propelled particles (NAGAI *et al.*, 2015). In computer science, it finds application in optimization (KENNEDY, 2006), robotics (HAYES; DORMIANI-TABATABAEI, 2002), and distributed computing (UPADHYAYA, 2013).

Gueleri *et al.* (2014) use an explicit dynamical system based on the flocking formation of certain living species like birds and fishes to solve SSL problems. Although the technique achieves considerable accuracy scores and has low computational cost, no mathematical analysis is performed on the system's characteristics. Moreover, the researchers argue that the richness of this self-organizing model is quite robust and may be used in other machine learning tasks.

Consequently, deeper studies of this dynamical system should be performed to get a better understanding of its potentialities and limitations.

1.3 Outline

In the next chapters, my major contributions to the area of machine learning based on complex networks and complex systems are exposed.

In Chapters 2 and 3, two papers on particle competition are presented. The first one introduces a new learning system for semi-supervised classification that employs collective and distributed dynamics. In the proposed process, particles compete for edges instead of vertices. As a result, more information is retrieved with the same time complexity. Moreover, the evolution function of the system is deterministic, and it is proved to be associated with the evolution of the expected values of the stochastic counterpart. Thus, the novel learning method is also stable, reducing the time cost of the model. The second paper studies the differences between edge- and vertex-centric approaches. It concludes that the edge-centric approach is preferred in many machine learning contexts.

In Chapters 4 and 5, contributions on network formation are shown. The first paper proposes a novel network formation technique that produces bipartite networks, called feature–sample networks. A system inspired by the collective random motion of particles is proposed to exploit the network and to solve positive-unlabeled (PU) learning problems. The second contribution consists in the enhancement of feature–sample networks by retrieving more information in a nonlinear manner.

In Chapter 6, a paper on the use of flocking systems to solve community detection problems is presented. A new community-detection method is developed. Such method employs a collective dynamics of aligning particles to decide whether each edge is intra- or inter-communities. The behavior of a collective system is characterized analytically, so the functioning, the potentialities and the shortcomings of the method are exposed.

Finally, in Chapter 7, I summarize my contributions to the field, discussing their impacts and limitations. Future research directions are addressed as well.

CHAPTER

2

NETWORK UNFOLDING MAP BY VERTEX-EDGE DYNAMICS MODELING

© 2018 IEEE. Reprinted, with permission, from Filipe A. N. Verri, Paulo R. Urio, Liang Zhao, "Network Unfolding Map by Vertex-Edge Dynamics Modeling", IEEE Transactions on Neural Networks and Learning Systems, 2018.

Contribution statement

F.A.N. Verri conceived the Labeled Component Unfolding system and devised the deterministic version. F.A.N. Verri and P.R. Urio designed the Network Unfolding Map learning model and analysed the results. F.A.N. Verri developed the mathematical analysis. P.R. Urio implemented both stochastic and deterministic models. F.A.N. Verri revised the implementations. P.R. Urio conducted the experiments that resulted in Figures 1, 3, 6, and 7, and Tables 2, 3, and 4. F.A.N. Verri performed the experiments that resulted in Figures 2, 4, and 5, and Tables 5 and 6. P.R. Urio, F.A.N. Verri, and L. Zhao contributed to the writing of the manuscript. L. Zhao supervised the project.

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

Network Unfolding Map by Vertex-Edge Dynamics Modeling

Filipe Alves Neto Verri, Paulo Roberto Urio, and Liang Zhao, *Senior member, IEEE*

***Abstract*—The emergence of collective dynamics in neural networks is a mechanism of the animal and human brain for information processing. In this paper, we develop a computational technique using distributed processing elements in a complex network, which are called particles, to solve semi-supervised learning problems. Three actions govern the particles' dynamics: generation, walking, and absorption. Labeled vertices generate new particles that compete against rival particles for edge domination. Active particles randomly walk in the network until they are absorbed by either a rival vertex or an edge currently dominated by rival particles. The result from the model evolution consists of sets of edges arranged by the label dominance. Each set tends to form a connected subnetwork to represent a data class. Although the intrinsic dynamics of the model is a stochastic one, we prove there exists a deterministic version with largely reduced computational complexity; specifically, with linear growth. Furthermore, the edge domination process corresponds to an unfolding map in such way that edges "stretch" and "shrink" according to the vertex-edge dynamics. Consequently, the unfolding effect summarizes the relevant relationships between vertices and the uncovered data classes. The proposed model captures important details of connectivity patterns over the vertex-edge dynamics evolution, in contrast to previous approaches which focused on only vertex or only edge dynamics. Computer simulations reveal that the new model can identify nonlinear features in both real and artificial data, including boundaries between distinct classes and overlapping structures of data.**

***Index Terms*—Complex networks, nonlinear dynamical systems, semi-supervised learning, particle competition.**

I. Introduction

SEMI-SUPERVISED learning (SSL) is one of the machine learning paradigms, which lies between the unsupervised and supervised learning paradigms. In SSL problems, both unlabeled and labeled data are taken into account in class or cluster formation and prediction processes [1], [2]. In real-world applications, we usually have partial knowledge on a given dataset. For example, we certainly do not know every movie actor except a few famous ones; in a large-scale social network, we just know some friends; in biological domain, we are far away from completely obtaining a figure of the functions of all genes, but we know the functions of some of them. Sometimes, although we have a complete or almost complete knowledge of a dataset, labeling it by hand is lengthy and expensive. So it is necessary to restrict the labeling scope. For these reasons, partially labeled datasets are often encountered. In this sense, supervised and unsupervised learning can be considered as extreme and special cases of semi-supervised learning. Many semi-supervised learning techniques have been developed, including generative models [3], discriminative models [4], clustering and labeling techniques [5], multi-training [6], low-density separation models [7], and graph-based methods [8]–[10]. Among the approaches listed above, graph-based SSL has triggered much attention. In this case, each data instance is represented by a vertex and is linked to other vertices according to a predefined affinity rule. The labels are propagated to the whole graph using a particular optimization heuristic [11].

Complex networks are large-scale graphs with nontrivial topology [12]. Such networks introduce a powerful tool to describe the interplay of topology, structure, and dynamics of complex systems [12], [13]. Therefore, they provide a groundbreaking mechanism to help us understand the behavior of many real systems. Networks also turn out to be an important mechanism for data representation and analysis [14]. Interpreting data sets as complex networks grant us to access the inter-relational nature of data items further. For this reason, we consider the network-based approach for SSL in this work. However, the above-mentioned network-based approach focuses on the optimization of the label propagation result and pays little attention to the detailed dynamics of the learning process itself. On the other hand, it is well-known that collective neural dynamics generate rich information, and such a redundant processing handles the adaptability and robustness of the learning process. Moreover, traditional graph-based techniques have high computational complexity, usually at cubic order [15]. A common strategy to overcome this disadvantage is using a set of sparse prototypes derived from the data [10]. However, such a sampling process usually loses information of the original data.

Taking into account the facts above, we study a new type of dynamical competitive learning mechanism in a complex network, called *particle competition*. Consider a network where several particles walk and compete to occupy as many vertices as possible while attempting to reject rival particles. Each particle performs a combined random and preferential walk by

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. Published version: http://ieeexplore.ieee.org/document/7762202/. DOI 10.1109/TNNLS.2016.2626341.

This research was supported by the São Paulo State Research Foundation (FAPESP), the Coordination for the Improvement of Higher Education Personnel (CAPES), and the Brazilian National Research Council (CNPq).

F.A.N. Verri and P.R. Urio are with the Institute of Mathematical and Computer Sciences (ICMC), University of São Paulo, São Carlos, SP, Brazil. Email: {filipeneto,urio}@usp.br.

L. Zhao is with Faculty of Philosophy, Sciences, and Letters at Ribeirão Preto (FFCLRP), University of São paulo, Ribeirão Preto, SP, Brazil. Email: zhao@usp.br.

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

choosing a neighbor vertex to visit. Finally, it is expected that each particle occupies a subset of vertices, called a community of the network. In this way, community detection is a direct result of the particle competition. The particle competition model was originally proposed in [16] and extended for the data clustering task in [17]. Later, it has been applied to semi-supervised learning [18], [19] where the particle competition is formally represented by a nonlinear stochastic dynamical system. In all the models mentioned above, the authors concern vertex dynamics—how each vertex changes its state (the level of dominance of each particle). Intuitively, vertex dynamics is a rough modeling of a network because each vertex can have several edges. A problem with the data analysis in this approach is the overlapping nature of the vertices, where a data item (a vertex in the networked form) can belong to more than one class. Therefore, it is interesting to know how each edge changes its state in the competition process to acquire detailed knowledge of the dynamical system.

In this paper, we propose a transductive semi-supervised learning model that employs a vertex–edge dynamical system in complex networks. In this dynamical system, namely Labeled Component Unfolding system, particles compete for *edges* in a network. Subnetworks are generated with the edges grouped by class dominance. Here, we call each subnetwork an *unfolding*. The learning model employs the unfoldings to classify unlabeled data. The proposed model offers satisfactory performance on semi-supervised learning problems, in both artificial and real dataset. Also, it has shown to be suitable for detecting overlapping regions of data points by simply counting the edges dominated by each class of particles. Moreover, it has low computational complexity order.

In comparison to the original particle competition models and other graph-based semi-supervised learning techniques, the proposed one presents the following salient features:

a) Particle competition dynamics occurs on nodes as well as on edges: The inclusion of the edge domination model can give us more detailed information to capture connectivity pattern of the input data. This is because there are much more edges than vertices even in a sparse network. Consequently, the proposed model has the benefit of granting essential information concerning overlapping vertices. Computer simulations show the proposed technique achieves a good classification accuracy and it is suitable for situations with a small number of labeled samples.

b) In the proposed model, particles are continuously generated and removed from the system: Such a feature contrasts to previous particle competition models that incorporate a preferential walking mechanism where particles tend to avoid rival particles. As a consequence, the number of active particles in the system varies over time. It is worth noting that the elimination of preferential walking mechanism largely simplifies the dynamical rules of particle competition model. Now, the new model is characterized by the competition of only random walking particles, which, in turn, permits us to find out an equivalent deterministic version. The original particle competition model is intrinsically stochastic. Then, each run may generate a different result. Consequently, it has high computational cost. In this work, we find out a deterministic system with running time independent of the number of particles, and we demonstrate that it is mathematically equivalent to the stochastic model. Moreover, the deterministic model has linear time order and ensures stable learning. In other words, the model generates the same output for each run with the same input. Furthermore, the system is simpler and easier to be understood and implemented. Thus, the proposed model is more efficient than the original particle competition model.

c) There is no explicit objective function: In classical graph-based semi-supervised learning techniques, usually, an objective function is defined for optimization. Such function considers not only the label information, but also the semi-supervised assumptions of smoothness, cluster, or manifold. In particle competition models, we do not need to define an objective function. Instead, dynamical rules which govern the time evolution of particles and vertices (or edges) are defined. Those dynamical rules mimic the phenomena observed in some natural and social systems, such as resource competition among animals, territory exploration by humans (or animals), election campaigns, etc. In other words, the particle competition technique is typically inspired by nature. In such kind of technique, we have focused on behavior modeling instead of objective modeling. Certain objectives can be achieved if the corresponding behavioral rules are properly defined. In this way, we may classify classical graph-based semi-supervised learning techniques as *objective-based design* and the particle competition technique as *behavior-based design*.

The remainder of this paper is organized as follows. The proposed particle competition system is studied in Section II. Our transductive semi-supervised learning model is represented in Section III. In Section IV, results of computer simulations are shown to assess the proposed model performance on both artificial and real-world datasets. Finally, Section V concludes this paper.

II. Labeled Component Unfolding system

In this section, we give an introduction to the Labeled Component Unfolding (LCU) system—a particle competition system for edge domination—explaining its basic design. Whenever pertinent, we go into detail for further clarification.

A. Overview

We consider a complex network expressed by a simple, unweighted, undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. If two vertices are considered similar, an edge connects them. The network contains $|\mathcal{V}| = l + u$ vertices that can be either labeled or unlabeled data points. The set $\mathcal{L} = \{v_1, \ldots, v_l\}$ contains the labeled vertices, where a vertex $v_i \in \mathcal{L}$ has a label $y_i \in \{1, \ldots, C\}$. We also use the terms *label* and *class* synonymously—if a vertex is labeled with c, we say this vertex belongs to class c. The set $\mathcal{U} = \{v_{l+1}, \ldots, v_{l+u}\}$ contains the unlabeled vertices. We suppose that $l \ll u$. Thus, we have that $\mathcal{L} \cap \mathcal{U} = \emptyset$ and $\mathcal{V} = \mathcal{L} \cup \mathcal{U}$. The network is represented by the adjacency matrix $A = (a_{ij})$ where $a_{ij} = a_{ji} = 1$ if v_i is connected to v_j. We denote (i, j) as the edge between vertices

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

v_i and v_j. For practical reasons, we consider a connected network, and there is at least one labeled vertex of each class.

In this model, particles are objects that flow within the network while carrying a label. Labeled vertices are *sources* for particles of the same class and *sinks* for particles of other classes. After a particle is released, it randomly walks the network. There is equal probability among adjacent vertices to be chosen as the next vertex to be visited by the particle. Consider that a particle is in v_i, it decides to move to v_j with probability

$$\frac{a_{ij}}{\deg v_i}$$

with $\deg v_i$ denoting the degree of v_i.

In each step, at the moment that a particle decides to move to a next vertex, it can be absorbed (removed from the system). If a particle is not absorbed, we say that it has survived and it remains active; and if it survives, then it continues walking. Otherwise, the particle is absorbed and ceases to affect the system. The absorption depends on the level of subordination and domination of a class against all other classes in the edges.

To determine the level of domination and subordination of each class in an edge, we take into account the active particles in the system. The *current directed domination* $\tilde{n}_{ij}^c(t)$ is the number of active particles belonging to class c that decided to move from v_i to v_j at time t and survived. Similarly, the *current relative subordination* $\tilde{\sigma}_{ij}^c$ is the fraction of active particles that do not belong to class c and have successfully passed through edge (i,j), *regardless of direction*, at time t. Mathematically, we define the latter as

$$\tilde{\sigma}_{ij}^c \coloneqq \begin{cases} 1 - \dfrac{\tilde{n}_{ij}^c + \tilde{n}_{ji}^c}{\sum_{q=1}^{C} \tilde{n}_{ij}^q + \tilde{n}_{ji}^q} & \text{if } \sum_{q=1}^{C} \tilde{n}_{ij}^q + \tilde{n}_{ji}^q > 0, \\ 1 - \dfrac{1}{C} & \text{otherwise.} \end{cases}$$

The survival of a particle depends on the current relative subordination of the edge and the destination vertex. If a particle decides to move into a sink, it will be absorbed with probability 1. If the destination vertex is not a sink, its survival probability is

$$1 - \lambda \tilde{\sigma}_{ij}^c(t)$$

where $\lambda \in [0,1]$ is a parameter for characterizing the competition level.

A source generates particles according to its degree and the current number of active particles in the system. Let $\tilde{n}^c(t)$ be the number of active particles belonging to class c in the system at time t, a source generates new particles if $\tilde{n}^c(t) < \tilde{n}^c(0)$.

Let $\mathcal{G}^c = \{v_i | v_i \in \mathcal{L} \text{ and } y_i = c\}$ be the set of sources for particles that belong to class c, the number of newly generated particles belonging to class c in v_i at time t follows the distribution

$$\begin{cases} \mathrm{B}(\tilde{n}^c(0) - \tilde{n}^c(t),\ \rho_i^c) & \text{if } \tilde{n}^c(0) - \tilde{n}^c(t) > 0, \\ \mathrm{B}(1,0) & \text{otherwise,} \end{cases}$$

where

$$\rho_i^c \coloneqq \begin{cases} \dfrac{\deg v_i}{\sum_{v_j \in \mathcal{G}^c} \deg v_j} & \text{if } v_i \in \mathcal{G}^c, \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathrm{B}(n,p)$ is a binomial distribution. In other words, if the number of active particles is fewer than the initial number of particles, $\tilde{n}^c(0)$, each source performs $\tilde{n}^c(0) - \tilde{n}^c(t)$ trials with probability ρ_i^c of generating a new particle.

Therefore, the expected number of new particles belonging to class c in v_i at time t is

$$\begin{cases} \rho_i^c \left(\tilde{n}^c(0) - \tilde{n}^c(t)\right) & \text{if } \tilde{n}^c(0) - \tilde{n}^c(t) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

We are interested in the total number of visits of particles of each class to each edge. Thus, we introduce the *cumulative domination* $\tilde{\delta}_{ij}^c(t)$ that is the total number of particles belonging to class c that passed through edge (i,j) up to time t. Mathematically, this is defined as

$$\tilde{\delta}_{ij}^c(t) \coloneqq \sum_{\tau=1}^{t} \tilde{n}_{ij}^c(\tau)\,. \tag{1}$$

Using the cumulative domination, we can group the edges by class domination. For each class c, the subset $\mathcal{E}^c(t) \subseteq \mathcal{E}$ is

$$\mathcal{E}^c(t) \coloneqq \left\{ (i,j) \middle| \arg\max_q \left(\tilde{\delta}_{ij}^q(t) + \tilde{\delta}_{ji}^q(t) \right) = c \right\}.$$

We define the subnetwork

$$G^c(t) \coloneqq (\mathcal{V}, \mathcal{E}^c(t)) \tag{2}$$

as the *unfolding* of network G according to class c at time t. We interpret the unfolding as a subspace with the most relevant relationships for a given class. We use the available information in these subnetworks for the study of overlapping regions and for semi-supervised learning.

B. An Illustrative Example

One iteration of the system's evolution is illustrated by Figure 1. The considered system contains 22 active particles at time t and 20 at time $t+1$. In an iteration, each particle moves to a neighbor vertex, without preference. The movement of a particle is indicated by an arrow. An interrupted line indicates an edge in which the coming particle is absorbed. A total of 6 particles are absorbed during this iteration, and the sources have generated 4 new particles.

At time t, for example, one of the red particles passing through edge $(1,3)$ is absorbed due to a current edge dominance of 0.5 in that edge (one red particle and one green particle). Conversely, all green particles that moved through edge $(5,7)$ remain active at time $t+1$. Since there is no rival particle (red particle) passing through this edge, the updated value of the current edge dominance is 1 and 0 for green and red classes, respectively.

In edge $(2,4)$, one red and two green particles chose to pass through. One green particle is absorbed without affecting the new current level of dominance. Since one particle of each class successfully passed through edge $(2,4)$, the new current level of dominance on this edge is 0.5. The same occurs for edge $(4,7)$ where no particles have passed through and, thus, the current level of dominance is set equally among all classes.

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

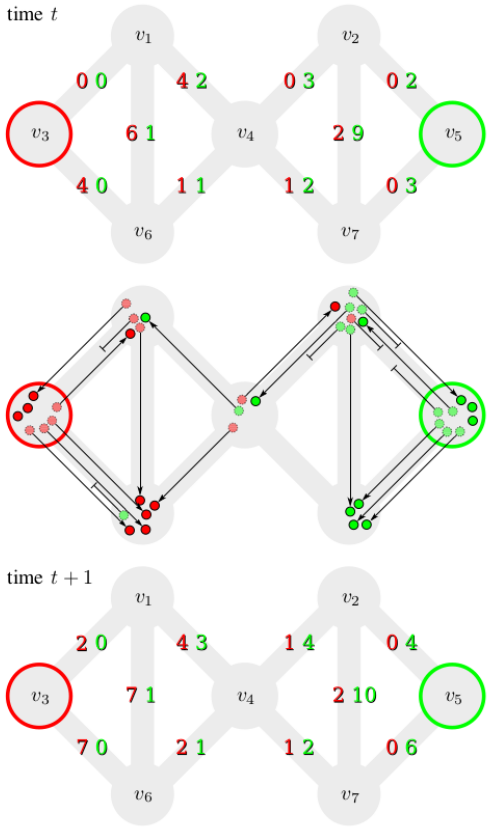


Fig. 1. Illustration of one iteration of the system's evolution. The network consists of 7 vertices and 10 edges; each color represents a label of a particle or a source. The first and the third networks depict the cumulative domination before and after the iteration. The cumulative domination is the number of visits of particles to an edge since the initial state. In the second network, particles are depicted in small circles. Active particles at time t are depicted in dashed borders, whereas active particles at time $t+1$ are in full borders. An arrow indicates a particle movement, while an interrupted line indicates that the particle has been absorbed when trying to move through an edge. Particles without an adjacent arrow are generated by the sources at time $t+1$.

In edges $(2,5)$ and $(3,6)$, particles have tried to move into a source of rival particles (sinks). These particles are absorbed independently from the current level of dominance.

Our edge-centric system can measure the overlapping nature of the v_4 by counting the edges dominated by each class, while a vertex-centric approach would have lost such information.

C. Mathematical Modeling

Formally, we define the Labeled Component Unfolding system as a dynamical system $\tilde{X}(t)$. The state of the system is

$$\tilde{X}(t) \coloneqq \begin{bmatrix} \tilde{\mathbf{n}}^c(t) = \left[\tilde{n}_i^c(t)\right]_i \\ \tilde{\Delta}^c(t) = \left(\tilde{\delta}_{ij}^c(t)\right)_{i,j} \end{bmatrix}, \tag{3}$$

where $\tilde{\mathbf{n}}^c(t)$ is a vector, and each element $\tilde{n}_i^c(t)$ is the number of active particles belonging to class c in v_i at time t. Furthermore, $\tilde{\Delta}^c(t)$ is a matrix whose elements $\tilde{\delta}_{ij}^c(t)$ are given by Equation (1).

Let $\tilde{g}_i^c(t)$ and $\tilde{a}_i^c(t)$ be, respectively, the number of particles generated and absorbed by v_i at time t. The evolution function $\tilde{\phi}$ of the dynamical system is

$$\tilde{\phi}: \begin{cases} \tilde{n}_i^c(t+1) = \tilde{n}_i^c(t) + \sum_j \left(\tilde{n}_{ji}^c(t+1) - \tilde{n}_{ij}^c(t+1)\right) \\ \qquad\qquad + \tilde{g}_i^c(t+1) - \tilde{a}_i^c(t+1)\,. \\ \tilde{\delta}_{ij}^c(t+1) = \tilde{\delta}_{ij}^c(t) + \tilde{n}_{ij}^c(t+1)\,. \end{cases}$$

Intuitively, the number $\tilde{n}_i^c$ of active particles that are in a vertex is the total number of particles arriving, $\tilde{n}_{ji}^c$, minus the number of particles leaving, $\tilde{n}_{ij}^c$, or being absorbed, $\tilde{a}_i^c$; additionally for labeled vertices, the number of generated particles, $\tilde{g}_i^c$. Moreover, to calculate the total number $\tilde{\delta}_{ij}^c$ of visits of particles to an edge, we simply add up the number $\tilde{n}_{ij}^c$ at each time. Values $\tilde{n}_{ij}^c$, $\tilde{g}_i^c$, and $\tilde{a}_i^c$ are obtained *stochastically* according to the dynamics of walking, absorption, and generation.

The initial state of the system is given by an arbitrary number $\tilde{n}_i^c(0)$ of initial active particles and

$$\begin{cases} \tilde{n}_{ij}^c(0) = 0\,, \\ \tilde{\delta}_{ij}^c(0) = 0\,. \end{cases}$$

To achieve the desirable network unfolding, it is necessary to average the results of several simulations of the system with a very large number of initial particles $\tilde{n}_i^c(0)$. Thus, the computational cost of such a simulation is very high. Conversely, we provide an alternative system $X(t)$ that achieves similar results in a *deterministic* manner. More details will follow.

D. Alternative Mathematical Modeling

Consider the dynamical system

$$X(t) \coloneqq \begin{bmatrix} \mathbf{n}^c(t) = \left[n_i^c(t)\right]_i \\ N^c(t) = \left(n_{ij}^c(t)\right)_{i,j} \\ \Delta^c(t) = \left(\delta_{ij}^c(t)\right)_{i,j} \end{bmatrix}, \tag{4}$$

where $\mathbf{n}^c(t)$ is a row vector whose elements $n_i^c(t)$ give the population of particles with label c in each vertex v_i at time t. These values are associated to the number of active particles $\tilde{n}_i^c$ of system $\tilde{X}$. The elements $n_{ij}^c(t)$ and $\delta_{ij}^c(t)$ of the sparse matrices $N^c(t)$ and $\Delta^c(t)$ are related to the *current directed domination*, $\tilde{n}_{ij}^c$, and the *cumulative domination*, $\tilde{\delta}_{ij}^c$, respectively. In other words, $n_{ij}^c(t)$ gives the number of particles of class c that moved from v_i to v_j at time t, while $\delta_{ij}^c(t)$ gives the total number up to time t.

The system X is a nonlinear Markovian dynamical system with the deterministic evolution function

$$\phi: \begin{cases} \mathbf{n}^c(t+1) = \mathbf{n}^c(t) \times P^c(X(t)) + \mathbf{g}^c(X(t)) \\ N^c(t+1) = \operatorname{diag} \mathbf{n}^c(t) \times P^c(X(t)) \\ \Delta^c(t+1) = \Delta^c(t) + N^c(t+1)\,, \end{cases} \tag{5}$$

where $\operatorname{diag} \mathbf{v}$ is a square matrix with the elements of vector $\mathbf{v}$ on the main diagonal and $\times$ stands for the vector-matrix product.

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

The function $P^c(X(t))$ of the system X at time t gives a square matrix whose elements are

$$p_{ij}^c(X(t)) \coloneqq \begin{cases} 0 & \text{if } v_j \in \mathcal{L} \text{ and } y_j \neq c\,, \\ \dfrac{a_{ij}}{\deg v_i}\left(1 - \lambda\sigma_{ij}^c(X(t))\right) & \text{otherwise,} \end{cases} \tag{6}$$

where

$$\sigma_{ij}^c(X(t)) \coloneqq \begin{cases} 1 - \dfrac{n_{ij}^c(t) + n_{ji}^c(t)}{S} & \text{if } S > 0, \\ 1 - \dfrac{1}{C} & \text{otherwise,} \end{cases}$$
$$\text{with } S = \sum_{q=1}^{C} n_{ij}^q(t) + n_{ji}^q(t). \tag{7}$$

Given that we know the initial state $X(0)$ of the system, the function $\mathbf{g}^c(X(t))$ of the system X at time t returns a row vector where the i-th element is

$$g_i^c(X(t)) \coloneqq \rho_i^c \max\{0,\ \mathbf{1}\cdot\mathbf{n}^c(0) - \mathbf{1}\cdot\mathbf{n}^c(t)\}\,, \tag{8}$$

where $\mathbf{1}$ is a row vector whose elements are 1, and $\cdot$ stands for the inner product between vectors.

The initial state of the system X is given by an arbitrary *population size*[1] $n_i^c(0)$ of initial active particles and

$$\begin{cases} n_{ij}^c(0) = 0\,, \\ \delta_{ij}^c(0) = 0\,. \end{cases}$$

If the initial number of particles in each vertex in system $\tilde{X}$ is proportional to the initial population size in system X, we provide evidence that the unfolding result tends to be the same for both systems—represented by Equation (3) and Equation (4), respectively—, as $\tilde{n}_i^c(0) \to \infty$ for all $c \in \{1,\ldots,C\}$ and $i \in \{1,\ldots,|\mathcal{V}|\}$.

E. Mathematical Analysis

In the previous subsections, we modeled two possibly equivalent systems, X and $\tilde{X}$. In this section, we present mathematical results that prove the equivalence of the two systems under certain assumptions.

Theorem 1. *Systems X and $\tilde{X}$ are asymptotically equivalent if the following conditions hold:*

$$\mathrm{E}\big[\tilde{\sigma}_{ij}^c(t)\big] \to 1 - \frac{\mathrm{E}\big[\tilde{n}_{ij}^c(t)\big] + \mathrm{E}\big[\tilde{n}_{ji}^c(t)\big]}{\sum_{q=1}^{C}\mathrm{E}\big[\tilde{n}_{ij}^q(t)\big] + \mathrm{E}\big[\tilde{n}_{ji}^q(t)\big]} \text{ and}$$
$$\mathrm{E}[\tilde{g}_i^c(t+1)] \to \rho_i^c \max\{0,\ \tilde{n}^c(0) - \mathrm{E}[\tilde{n}^c(t)]\} \text{ as}$$
$$\tilde{n}_i^c(0) \to \infty,$$

for all $i, j \in \mathcal{V}$, $t > 0$, and $c \in \{1,\ldots,C\}$, we have

$$n_i^c(t) = \kappa\mathrm{E}[\tilde{n}_i^c(t)]\,,\ n_{ij}^c(t) = \kappa\mathrm{E}\big[\tilde{n}_{ij}^c(t)\big]\,, \text{ and}$$
$$\delta_{ij}^c(t) = \kappa\mathrm{E}\Big[\tilde{\delta}_{ij}^c(t)\Big]\,,$$

for some $k > 0$ constant.

In order to prove Theorem 1, we study the following mechanisms of the particle competition system:

1) Particle motion and absorption: In the proposed system, each particle moves independently from the others. Particle's movement through an edge affects the absorption of rival particles only in the next iteration. Such conditions are favorable to naturally regard the system's evolution in terms of the distribution of particles over the network. Next, we present a formal model for particle movement.

Let $\mathrm{I}_{ij}(p,\ t+1)$ be a discrete random variable that is 1 if particle p was in v_i at time t and moved into v_j at time $t+1$; and it is 0 otherwise. Since each particle in a vertex moves independently, we can write this probability in terms of a particle's class; that is, $\mathrm{I}_{ij}^c(t+1) = \mathrm{I}_{ij}(p,\ t+1)$ for any particle p that belongs to class c and is in v_i at time t.

The probability $\Pr\big[\mathrm{I}_{ij}^c(t+1) = 1\big]$ is affected by the movement decision of a particle and whether it was absorbed after the decision. By formulation, in dynamical system $\tilde{X}$ the conditional probability, given that $\tilde{\sigma}_{ij}^c(t) = \xi$, is

$$\Pr\big[\mathrm{I}_{ij}^c(t+1) = 1 \big| \tilde{\sigma}_{ij}^c(t) = \xi\big] = \begin{cases} 0 & \text{if } v_j \in \mathcal{L} \text{ and } y_j \neq c\,, \\ \dfrac{a_{ij}}{\deg v_i}\left(1 - \lambda\cdot\xi\right) & \text{otherwise.} \end{cases}$$

That is, when a particle tries to move into a sink, the survival probability is zero. Otherwise, a particle only reaches v_j if it chooses to move into the vertex and it is not absorbed.

Let $f_{\tilde{\sigma}_{ij}^c(t)}$ be the probability density function of the random variable $\tilde{\sigma}_{ij}^c$. Hence, the probability $\Pr\big[\mathrm{I}_{ij}^c(t+1) = 1\big]$ is

$$\begin{aligned} &\int_{-\infty}^{\infty} \Pr\big[\mathrm{I}_{ij}^c(t+1) = 1 \big| \tilde{\sigma}_{ij}^c(t) = \xi\big]\, f_{\tilde{\sigma}_{ij}^c(t)}(\xi)\,\mathrm{d}\xi \\ &\qquad = \int_{-\infty}^{\infty} \frac{a_{ij}}{\deg v_i}\left(1 - \lambda\cdot\xi\right) f_{\tilde{\sigma}_{ij}^c(t)}(\xi)\,\mathrm{d}\xi \\ &= \frac{a_{ij}}{\deg v_i}\left(\int_{-\infty}^{\infty} f_{\tilde{\sigma}_{ij}^c(t)}(\xi)\,\mathrm{d}\xi - \lambda\int_{-\infty}^{\infty}\xi f_{\tilde{\sigma}_{ij}^c(t)}(\xi)\,\mathrm{d}\xi\right) \\ &\qquad = \frac{a_{ij}}{\deg v_i}\left(1 - \lambda\mathrm{E}\big[\tilde{\sigma}_{ij}^c(t)\big]\right), \end{aligned}$$

if $v_j \in \mathcal{U}$ or $v_j \in \mathcal{L} \wedge y_j \neq c$. Otherwise, it is zero.

Furthermore, $\tilde{\sigma}_{ij}^c$ is convex with fixed values of $\tilde{n}_{ij}^q, \tilde{n}_{ji}^q$ for all $q \neq c$. Thus, with the Jensen's inequality [20], we have

$$\mathrm{E}\big[\tilde{\sigma}_{ij}^c(t)\big] \geq 1 - \frac{\mathrm{E}\big[\tilde{n}_{ij}^c(t)\big] + \mathrm{E}\big[\tilde{n}_{ji}^c(t)\big]}{\sum_{q=1}^{C}\mathrm{E}\big[\tilde{n}_{ij}^q(t)\big] + \mathrm{E}\big[\tilde{n}_{ji}^q(t)\big]}. \tag{9}$$

2) Particle generation: In dynamical system $\tilde{X}$ the expected number of particles belonging to class c generated at v_i at time $t+1$ is

$$\mathrm{E}[\tilde{g}_i^c(t+1)] = \sum_{\eta=0}^{\infty}\mathrm{E}[\tilde{g}_i^c(t+1)|\tilde{n}^c(t) = \eta]\Pr[\tilde{n}^c(t) = \eta]\,.$$

The conditional expectation $\mathrm{E}[\tilde{g}_i^c(t+1)|\tilde{n}^c(t) = \eta]$ is, by formulation,

$$\rho_i^c \cdot \max\{0,\ \tilde{n}^c(0) - \eta\}\,,$$

[1] In system X, vector $\mathbf{n}^c(t)$ describes the quantity of particles in each vertex. Since X has multiplicative scaling behavior, $\mathbf{n}^c(t)$ is not necessarily composed only of integer values; $\mathbf{n}^c(t)$ values can be a discrete distribution of particles. See Section II-E5 for more details.

and thus, $\mathrm{E}[\tilde{g}_i^c(t+1)]$ is

$$\rho_i^c \sum_{\eta=0}^{\infty} \max\{0,\ \tilde{n}^c(0)-\eta\} \Pr[\tilde{n}^c(t)=\eta] = \rho_i^c \,\mathrm{E}[\max\{0,\ \tilde{n}^c(0)-\tilde{n}^c(t)\}]\,.$$

Since $\max\{0,\ x\}$ is convex for all $x \in \mathbb{R}$ and according to Jensen's inequality, we have

$$\mathrm{E}[\tilde{g}_i^c(t+1)] \geq \rho_i^c \max\{0,\ \tilde{n}^c(0)-\mathrm{E}[\tilde{n}^c(t)]\}\,. \tag{10}$$

3) Expected edge domination: At the beginning of system $\tilde{X}$ we have

$$\tilde{\delta}_{ij}^c(0) = 0$$

and, for $t \geq 0$,

$$\mathrm{E}\Big[\tilde{\delta}_{ij}^c(t+1)\Big] = \mathrm{E}\Big[\tilde{\delta}_{ij}^c(t)+\tilde{n}_{ij}^c(t+1)\Big] = \mathrm{E}\Big[\tilde{\delta}_{ij}^c(t)\Big] + \mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big]\,. \tag{11}$$

Given that $\tilde{n}_i^c(t)=\eta$ is known and since each particle in a vertex moves independently, the number of particles that successfully reaches v_j at time $t+1$ is

$$\tilde{n}_{ij}^c(t+1) = \sum_{k=1}^{\eta} \mathrm{I}_{ij}(p_k,\ t+1)\,,$$

where p_k is a particle that belongs to class c and is in v_i. Then, the expected value $\mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big]$ is

$$\begin{aligned}
&\sum_{\eta=0}^{\infty} \mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big|\tilde{n}_i^c(t)=\eta\big] \Pr[\tilde{n}_i^c(t)=\eta] \\
&\quad= \sum_{\eta=0}^{\infty} \Pr[\tilde{n}_i^c(t)=\eta] \cdot \sum_{k=1}^{\eta} \mathrm{E}[\mathrm{I}_{ij}(p_k,\ t+1)] \\
&\quad= \sum_{\eta=0}^{\infty} \Pr[\tilde{n}_i^c(t)=\eta] \cdot \sum_{k=1}^{\eta} \Pr[\mathrm{I}_{ij}(p_k,\ t+1)=1] \\
&\quad= \sum_{\eta=0}^{\infty} \eta \Pr[\tilde{n}_i^c(t)=\eta] \cdot \Pr\big[\mathrm{I}_{ij}^c(t)=1\big]\,.
\end{aligned}$$

Finally,

$$\mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big] = \mathrm{E}[\tilde{n}_i^c(t)]\Pr\big[\mathrm{I}_{ij}^c(t)=1\big] \tag{12}$$

for all $t \geq 0$, $c=\{1,\ldots,C\}$, and $i,j \in \{1,\ldots,|\mathcal{V}|\}$.

4) Expected number of particles: We know the number of particles at the beginning of system $\tilde{X}(t)$, so

$$\mathrm{E}[\tilde{n}_i^c(0)] = \tilde{n}_i^c(0)$$

and, for all $t \geq 0$, the expected value $\mathrm{E}[\tilde{n}_i^c(t+1)]$ is

$$\mathrm{E}[\tilde{n}_i^c(t)] + \sum_j \Big(\mathrm{E}\big[\tilde{n}_{ji}^c(t+1)\big] - \mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big]\Big) + \mathrm{E}[\tilde{g}_i^c(t+1)] - \mathrm{E}[\tilde{a}_i^c(t+1)]\,.$$

However, the expected number of particles that were absorbed in v_i is the expected number of particles in v_i minus the expected number of particles that survived when moving away. Thus, $\mathrm{E}[\tilde{n}_i^c(t+1)]$ can be written as

$$\mathrm{E}[\tilde{n}_i^c(t)] + \sum_j \Big(\mathrm{E}\big[\tilde{n}_{ji}^c(t+1)\big] - \mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big]\Big) + \mathrm{E}[\tilde{g}_i^c(t+1)] - \left(\mathrm{E}[\tilde{n}_i^c(t)] - \sum_j \mathrm{E}\big[\tilde{n}_{ij}^c(t+1)\big]\right).$$

And, finally

$$\mathrm{E}[\tilde{n}_i^c(t+1)] = \sum_j \mathrm{E}\big[\tilde{n}_{ji}^c(t+1)\big] + \mathrm{E}[\tilde{g}_i^c(t+1)]\,, \tag{13}$$

for all $t \geq 0$, $c=\{1,\ldots,C\}$, and $i \in \{1,\ldots,|\mathcal{V}|\}$.

5) Scale invariance: The unfolding $G^c(t)$ from system X is invariant under real positive multiplication of the row vector $\mathbf{n}^c(0)$. In order to prove this property, consider the following lemma.

Lemma 1. *System X has positive multiplicative scaling behavior of order 1. Given an arbitrary initial state X_0 of the system X, it means that*

$$X(t) = X_t \mid X(0) = X_0 \iff X(t) = \kappa X_t \mid X(0) = \kappa X_0 \tag{14}$$

for all $t > 0$ and $\kappa > 0$.

Proof of Lemma 1: First, we show that the functions P^c are invariant to parameter scaling. Given an arbitrary system state $X(t)=X_t$ and $\kappa > 0$,

$$p_{ij}^c(\kappa X_t) = p_{ij}^c(X_t) = \begin{cases} 0 & \text{if } v_j \in \mathcal{L} \text{ and } y_j \neq c\,, \\ \dfrac{a_{ij}}{\deg v_i}\left(1-\lambda\sigma_{ij}^c(X_t)\right) & \text{otherwise,} \end{cases}$$

since the term σ_{ij}^c can be either

$$\begin{aligned}
\sigma_{ij}^c(\kappa X_t) &= 1 - \frac{\kappa n_{ij}^c(t;X_t) + \kappa n_{ji}^c(t;X_t)}{\sum_{q=1}^{C} \kappa n_{ij}^q(t;X_t) + \kappa n_{ji}^q(t;X_t)} \\
&= 1 - \frac{n_{ij}^c(t;X_t) + n_{ji}^c(t;X_t)}{\sum_{q=1}^{C} n_{ij}^q(t;X_t) + n_{ji}^q(t;X_t)} = \sigma_{ij}^c(X_t)
\end{aligned}$$

or

$$\sigma_{ij}^c(\kappa X_t) = \sigma_{ij}^c(X_t) = 1 - \frac{1}{C}.$$

Now, consider two arbitrary initial states,

$$X_0 = \begin{cases} n_i^c(0) = \eta_i \\ n_{ij}^c(0) = 0 \\ \delta_{ij}^c(0) = 0 \end{cases} \quad \text{and} \quad \kappa X_0 = \begin{cases} n_i^c(0) = \kappa\eta_i \\ n_{ij}^c(0) = 0 \\ \delta_{ij}^c(0) = 0 \end{cases},$$

for all i,j,c. We have that,

$$\begin{aligned}
n_{ij}^c(1;\kappa X_0) &= n_{ij}^c(0;\kappa X_0) p_{ij}^c(\kappa X_0) \\
&= \kappa n_{ij}^c(0;X_0) p_{ij}^c(X_0) = \kappa n_{ij}^c(1;X_0)\,,
\end{aligned}$$

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

$$n_i^c(1;\kappa X_0) = \sum_j n_{ji}^c(1;\kappa X_0) + g_i^c(\kappa X_0)$$
$$= \kappa \sum_j n_{ji}^c(1;X_0) + 0 = \kappa n_i^c(1;X_0),$$

and

$$\delta_{ij}^c(1;\kappa X_0) = \delta_{ij}^c(0;\kappa X_0) + n_{ij}^c(1;\kappa X_0) =$$
$$0 + \kappa n_{ij}^c(1;X_0) = \kappa \delta_{ij}^c(1;X_0)$$

Thus, Relation (14) holds true for $t = 1$.

Assuming that Relation (14) holds true for some time t, we show that the relation holds true for $t+1$:

$$n_{ij}^c(t+1;\kappa X_0) = n_{ij}^c(t;\kappa X_0) p_{ij}^c(\kappa X_t)$$
$$= \kappa n_{ij}^c(t;X_0) p_{ij}^c(X_t) = \kappa n_{ij}^c(t+1;X_0)$$

$$n_i^c(t+1;\kappa X_0) = \sum_j n_{ji}^c(t+1;\kappa X_0) + g_i^c(\kappa X_t)$$
$$= \kappa \sum_j n_{ji}^c(t+1;X_0) + \kappa g_i^c(X_t) = \kappa n_i^c(t+1;X_0)$$

since

$$g_i^c(\kappa X_t) = \rho_i^c \max\{0,\ \kappa \mathbf{1} \cdot \mathbf{n}^c(0;X_0) - \kappa \mathbf{1} \cdot \mathbf{n}^c(t;X_t)\},$$

and

$$\delta_{ij}^c(t+1;\kappa X_0) = \delta_{ij}^c(t;\kappa X_0) + n_{ij}^c(t+1;\kappa X_0)$$
$$= \kappa \delta_{ij}^c(t;X_0) + \kappa n_{ij}^c(t+1;X_0) = \kappa \delta_{ij}^c(t+1;X_0)$$

So Relation (14) indeed holds true for $t+1$.

Since both the basis and the inductive step have been performed, by mathematical induction, the lemma is proved for all $t \geq 0$ natural. ■

Finally, using these studies, we may prove the theorem.

Proof of Theorem 1: By Equations (11)–(13), we have

$$\begin{cases} \mathrm{E}[\tilde{n}_i^c(t+1)] = \sum_j \mathrm{E}\left[\tilde{n}_{ji}^c(t+1)\right] + \mathrm{E}[\tilde{g}_i^c(t+1)], \\ \mathrm{E}\left[\tilde{n}_{ij}^c(t+1)\right] = \mathrm{E}[\tilde{n}_i^c(t)] \Pr\left[\mathrm{I}_{ij}^c(t) = 1\right], \\ \mathrm{E}\left[\tilde{\delta}_{ij}^c(t+1)\right] = \mathrm{E}\left[\tilde{\delta}_{ij}^c(t)\right] + \mathrm{E}\left[\tilde{n}_{ij}^c(t+1)\right], \end{cases}$$

which is system X assuming that Inequalities (9) and (10) tend to equality when there is a large number of particles and $\kappa \tilde{n}_i^c(0) = n_i^c(0)$, for any $k > 0$ constant (scale invariance property). ■

Remark 1. *Even if the convergence of Inequalities* (9) *and* (10) *are not true, another property that possibly makes the two systems equivalent is the compensation over time. At the beginning, both systems are equal; however, in the next iteration both absorption probability* (9) *and generated particles* (10) *are underestimated. Consequently, particles that have survived may compensate the ones that were not generated. Furthermore, the lower the number of absorbed particles in an iteration, the higher the absorption probability in the next iteration. Likewise, the lower the number of generated particles in an iteration, the higher is the expected number of new particles in the next iteration.*

III. Semi-supervised Learning by Labeled Component Unfolding

Unfoldings generated by LCU system are incorpored in a semi-supervised learning model. Consider two sets $\mathcal{X}_{\text{labeled}} = \{x_1, \ldots, x_l\}$ and $\mathcal{X}_{\text{unlabeled}} = \{x_{l+1}, \ldots, x_{l+u}\}$ such that $x_i \in \mathbb{R}^D$ for all i. Each data point $x_i \in \mathcal{X}_{\text{labeled}}$ is associated to a label $y_i \in \{1, \ldots, C\}$. In the semi-supervised learning setting, our goal is to correctly assign existing labels to the unlabeled data $\mathcal{X}_{\text{unlabeled}}$.

In short, the proposed learning model has three steps: *a*) a network is constructed based on a dataset composed of feature vectors, where vertices represent data points, and edges represent similarity relationship; *b*) LCU system is applied to obtain the unfoldings, that is, a distinct set of edges for each class of the dataset; and *c*) infer labels for every data point in $\mathcal{X}_{\text{unlabeled}}$.

Next, each step of the proposed learning model is presented in detail. Further to the model's algorithm description, its computational complexity analysis is also presented.

Since the proposed dynamical system takes place on a complex network, the original dataset needs to be represented in a network structure. Therefore, the first step of our learning model is to obtain a network representation. Each data point is associated to a single vertex of the network. Moreover, the network must be sparse, undirected, and unweighted. Labeled vertices correspond to the set of points in $\mathcal{X}_{\text{labeled}}$, and unlabeled vertices to the set of points in $\mathcal{X}_{\text{unlabeled}}$. Two vertices are connected by an edge if they have a relationship of similarity, which is determined by some metric or by the particular problem. Any graph construction method that satisfies such conditions may be used in this step. The *k*-Nearest Neighbor (*k*-NN) graph construction method is one of them.

The second step is to run system X defined by Equation (5) using the constructed complex network as its input. Two conditions are satisfied on the system initialization. First, no class should be privileged. Second, during the first iterations, all particles should be able to flow within the network with a small probability of absorption. Thus, the initial conditions of the system, for all i, j, and $c \in \{1, \ldots, C\}$, are

$$\begin{aligned} n_i^c(0) &= \frac{\deg v_i}{2\,|\mathcal{E}|}, \\ n_{ij}^c(0) &= 0, \\ \delta_{ij}^c(0) &= 0. \end{aligned} \tag{15}$$

Since there are always particles in the system, the iteration of system X should be stopped if the time limit has been reached. The time limit parameter τ controls the maximum number of iterations of the system.

At the last step, the networks $G^c(\tau)$ are used for vertex classification. We assign a label $y_j \in \{1, \ldots, C\}$ for each unlabeled vertex $v_j \in \mathcal{U}$, with the information provided by the networks G^c. Label y_j is assigned based on the density of edges in its neighborhood. Formally, the label for v_j is

$$y_j = \underset{c \in \{1,\ldots,C\}}{\arg\max} \left|\mathcal{E}(\mathcal{N}_{c,j})\right|, \tag{16}$$

Algorithm 1 Semi-supervised Learning by LCU.

1: **function** CLASSIFIER($\mathcal{X}_{\text{labeled}}$, $\mathcal{X}_{\text{unlabeled}}$, λ, τ)
2: $G \leftarrow$ BUILDNETWORK($\mathcal{X}_{\text{labeled}}$, $\mathcal{X}_{\text{unlabeled}}$)
3: subnetworks $\leftarrow$ UNFOLD(G, λ, τ)
4: **return** CLASSIFY($\mathcal{X}_{\text{unlabeled}}$, subnetworks)
5: **end function**

Algorithm 2 Labeled Component Unfolding system.

1: **function** UNFOLD(G, λ, τ)
2: **for** $c \in \{1, \ldots, C\}$ **do**
3: $\mathbf{n}^c \leftarrow n_0(G, c)$ ▷ Equation (15)
4: $N^c \leftarrow N_0(G, c)$
5: $\Delta^c \leftarrow \Delta_0(G, c)$
6: **end for**
7: **for** $t \in \{1, \ldots, \tau\}$ **do**
8: **for** $c \in \{1, \ldots, C\}$ **do**
9: $P^c \leftarrow P(G, N^1, \ldots, N^C, \lambda)$ ▷ Equation (6)
10: $\mathbf{g}^c \leftarrow g(G, \mathbf{n}^c, t)$ ▷ Equation (8)
11: $N^c \leftarrow \operatorname{diag} \mathbf{n}^c \times P^c$
12: $\mathbf{n}^c \leftarrow \mathbf{n}^c \times P^c + \mathbf{g}^c$
13: $\Delta^c \leftarrow \Delta^c + N^c$
14: **end for**
15: **end for**
16: **return** SUBNETWORKS(G, Δ^c) ▷ Equation (2)
17: **end function**

where $\mathcal{N}_{c,j}$ is the neighborhood of v_j in the unfolding $G^c(\tau)$. We denote the number of edges in this neighborhood as $|\mathcal{E}(\mathcal{N}_{c,j})|$.

A. Algorithm

Algorithm 1 summarizes the steps of our learning model. The algorithm accepts the labeled dataset $\mathcal{X}_{\text{labeled}}$, the unlabeled dataset $\mathcal{X}_{\text{unlabeled}}$, and 2 user-defined parameters—the competition (λ) parameter of the system X and the time limit parameter (τ). Moreover, it is necessary to choose a network formation technique.

The first step of the learning model is mapping the original vector-formed data to a network using a chosen network formation technique. Afterward, we unfold the network as described in Algorithm 2. This algorithm iterates the LCU system to produce one subnetwork for each class. Steps 2–6 initialize the system state as indicated in Equation (15). Steps 7–15 iterate the system until τ using the evolution function ϕ (5). Step 16 calculates and returns the unfoldings for each class. Back to Algorithm 1, by using the produced unfoldings, the unlabeled data are classified as described in Equation (16).

B. Computational Complexity and Running Time

Here, we provide the computational complexity analysis step by step.

The construction of the complex network from the input dataset depends on the chosen method. Since $|\mathcal{V}| = |\mathcal{X}_{\text{labeled}}| + |\mathcal{X}_{\text{unlabeled}}|$ is the number of data samples. The k-NN method, for example, has complexity order of $\mathcal{O}(D\,|\mathcal{V}|\log|\mathcal{V}|)$ using multidimensional binary search tree [21].

TABLE I
TIME COMPLEXITY OF COMMON GRAPH-BASED TECHNIQUES DISREGARDING THE GRAPH CONSTRUCTION STEP

Algorithm	Time Complexity
Transductive SVM [7]	$C\,\lvert\mathcal{V}\rvert^3$
Local and Global Consistency [22]	$\lvert\mathcal{V}\rvert^3$
Large Scale Transductive SVM [23]	$C\,\lvert\mathcal{V}\rvert^2$
Dynamic Label Propagation [24]	$\lvert\mathcal{V}\rvert^2$
Label Propagation [25]	$\lvert\mathcal{V}\rvert^2$
Original Particle Competition [17]	$C^2\,\lvert\mathcal{V}\rvert + C\,\lvert\mathcal{E}\rvert$
Labeled Component Unfolding	$C\,\lvert\mathcal{V}\rvert + C\,\lvert\mathcal{E}\rvert$
Minimum Tree Cut [26]	$\lvert\mathcal{V}\rvert$

The second step is running system X defined by Equation (5). Using sparse matrices, the system initialization, steps 2–6 of Algorithm 2, has complexity order of $\mathcal{O}(C\,|\mathcal{V}| + C\,|\mathcal{E}|)$. The system iteration calculates τC times the evolution function ϕ (5) represented in steps 8–14. The time complexity of each part of the system evolution is presented below.

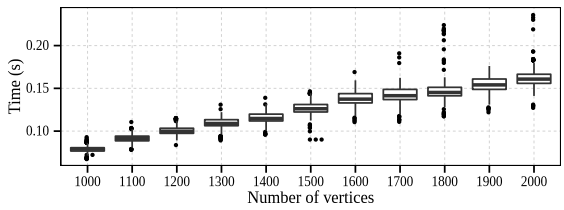
- Step 9, computation of the matrix P^c. This matrix has $|\mathcal{E}|$ non-zero entries. It is necessary to calculate σ_{ij}^c for each non-zero entry. Hence, this step has complexity order of $\mathcal{O}(C\,|\mathcal{E}|)$. However, the denominator of Equation (7) is the same for all values of c.
- Step 10, computation of the vector $\mathbf{g}^c$. This vector has $|\mathcal{L}|$ non-zero entries. It is also necessary to calculate the total number of particles in the system. So, this calculation has time complexity order of $\mathcal{O}(|\mathcal{L}| + |\mathcal{V}|)$.
- Step 11, computation of the matrix N^c. The multiplication between a diagonal matrix and a sparse matrix with $|\mathcal{E}|$ non-zero entries has time complexity order of $\mathcal{O}(|\mathcal{E}|)$.
- Step 12, computation of the vector $\mathbf{n}^c$. Suppose that $\langle k\rangle$ is the average vertex degree of the input network; it follows that this can be performed in $\mathcal{O}(|\mathcal{V}|\,\langle k\rangle) = \mathcal{O}(|\mathcal{E}|)$.
- Step 13, computation of the matrix Δ^c. This sparse matrix summation has complexity order of $\mathcal{O}(|\mathcal{E}|)$.

After the system evolution, the unfolding process performs $\mathcal{O}(C\,|\mathcal{E}|)$ operations. Thus, the total time complexity order of the system simulation is $\mathcal{O}(\tau C\,|\mathcal{E}| + \tau C\,|\mathcal{V}|)$. However, the value of τ is fixed and the value of C is usually very small.

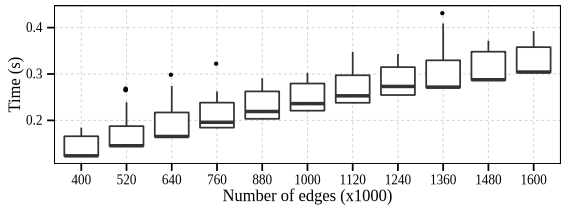
The vertex labeling step is the last step of the learning model. The time complexity of this step depends on the calculation of the number of edges in the neighborhood of each unlabeled vertex in each unfolding. It can be efficiently calculated by using one step of a breadth-first search in G^c. Hence, the order of the average-time complexity is $\mathcal{O}\big(C\,|\mathcal{U}|\,\langle k\rangle^2\big) \approx \mathcal{O}(C\,|\mathcal{E}|)$.

In summary, considering the discussion above, our learning model runs in $\mathcal{O}(D\,|\mathcal{V}|\log|\mathcal{V}| + C\,|\mathcal{E}| + C\,|\mathcal{V}|)$ including the transformation from vector-based dataset to a network. Table I compares the time complexity of common graph-based techniques disregarding the graph construction step. Only the proposed LCU method and Minimum Tree Cut [26] have linear time, though the latter must either receive or construct a spanning tree. Consequently, the Minimum Tree Cut has a performance similar to the scalable version of traditional

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341



(a)



(b)

Fig. 2. Running time in seconds of iterations of the system in random networks. (a) The input networks have 400 000 edges and many different numbers of vertices. (b) 2 000 vertices and many different numbers of edges.

algorithms, such as those using subsampling practices.

Figure 2 depicts the running time of a single iteration of the system varying the number of vertices and edges, respectively. With 10 independent runs, we measure the time for 30 iterations, totalizing 300 samples for each network size. We set $\lambda = 1$, two classes, and 5% of labeled vertices. Experiments were run on an Intel® Core™ i7 CPU 860 @ 2.80GHz with 16 GB RAM memory DDR3 @ 1333 MHz. This experiment shows that the system runs in linear time as a function of the number of vertices and edges, which conforms our theoretical analysis.

IV. Computer Simulations

To study the stochastic system $\tilde{X}$ and the deterministic version X, we present experimental analyses that concern their equivalence. Additionally, we study the meaning of the parameters of our learning model. After that, we evaluate the model performance using both artificial and synthetic datasets. Then, we show the unfolding process and the learning model on synthetic data. Finally, we present the simulation results for a well-known benchmark dataset and for a real application on human activity and handwritten digits recognition.

A. Experimental Analysis

In this section, we present an experiment that assesses the equivalence between the unfolding results of both systems with an increasing initial number of particles in system $\tilde{X}$.

The networks used for the analysis are generated by the following model: a complex network $G(\mathbf{y}, m, p)$ is constructed given a labeled vector $\mathbf{y}$, a number $m > 0$ of edges by vertex, and a weight $p \in [0, 1]$ that controls the preferential attachment between vertices of different classes. The resulting network contains $|\mathbf{y}|$ vertices. For each v_i, m edges are randomly connected, with replacement. If $y_i = y_j$, the preferential attachment weight is $1 - p$; otherwise, the weight is p. The parameter p is proportional to the overlap between classes.

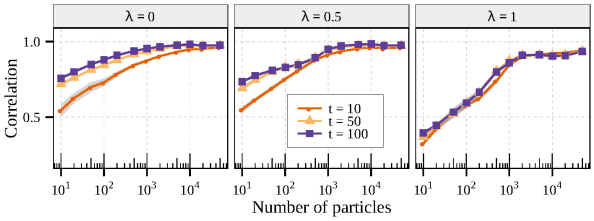


Fig. 3. Proportionality simulation. Lines are the correlation measure between the cumulative domination matrices of systems X and $\tilde{X}$, varying the initial number of active particles. Values close to 1 indicate that the cumulative domination matrices of both systems tend to be proportional.

If there exists a positive constant κ such that

$$\tilde{\delta}^c_{ij}(t) = \kappa \delta^c_{ij}(t),$$

both systems generate the same unfoldings. To assess this proportionality, both systems are simulated in 10 different networks $G(\mathbf{y}, 3, 0.05)$, with $|\mathbf{y}| = 200$ vertices arranged in two classes. The system's parameter is discretized in $\lambda = \{0, 0.5, 1\}$. Varying the total number of initial particles, we set $\tilde{n}^c_i(0) = n^c_i(0) \sim \deg v_i$ for all $c \in \{0, \ldots, C\}$ and $i \in \{1, \ldots, |\mathcal{V}|\}$.

We consider the correlation between the cumulative domination matrices of systems X and $\tilde{X}$. If the two matrices are proportional, then they must be correlated. Values of correlation close to 1 indicate the cumulative domination matrices are proportional. In Figure 3, the correlation is depicted. As the number of initial particles increases, the correlation approaches 1. This result suggests that both systems generate the same unfolding when the number of initial particles grows to infinity.

B. Parameter Analysis

The LCU model has two parameters apart from the network construction. In this section, we discuss their meaning. To do so, the learning model is applied in synthetic datasets whose data items are sampled from a three dimensional knot torus $\mathbf{v}(\theta)$ with parametric curve

$$x(\theta) = r(\theta)\cos 3\theta,$$
$$y(\theta) = r(\theta)\sin 3\theta,$$
$$z(\theta) = -\sin 4\theta,$$

where $\theta \in [0, 2\pi]$ and $r(\theta) = 2 + \cos 4\theta$.

We sampled 500 data items uniformly along the possible values of θ. We randomly split the data items from 2 to 10 classes so that the samples with adjacent θ belongs to the same class. We also added to each sample a random noise in each dimension with distribution $\mathcal{N}(0, \sigma)$ with $\sigma = 0.25$ and 0.35. Figure 4 depicts an example of the dataset with 4 classes with and without noise. Since the dataset has a complex form, a small change of parameter value may generate different results. Therefore, it is suitable to study the sensitivity of parameters.

We run the LCU model with parameters $\lambda \in \{0.25, 0.5, 0.75, 1\}$ and $\tau = 500$. Finally, 30 unbiased sets

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

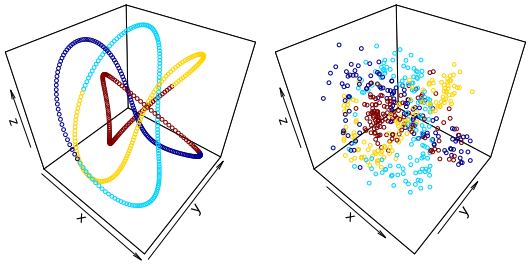


Fig. 4. Three dimensional knot torus dataset with 500 samples without noise (left-hand side) and with noise (right-hand side). Colors are the classes.

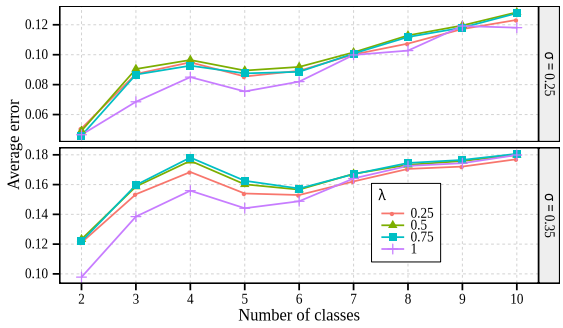


Fig. 5. Average error of the proposed model for different numbers of classes in the problem. Colors and shapes indicate the values of parameter λ.

of 40 labeled points are employed. The k-NN is used for the network construction with $k = \{4, 5, \ldots, 10\}$.

Below, we discuss each parameter of the model.

1) Discussion about the network construction parameter: In our model, the input network must be simple (between any pair of vertices there must exist at most one edge), unweighted, undirected, and connected. Besides these requirements, two vertices must be connected if their data items are considered similar enough to the particular problem. In our experiments, we use k-NN graph with Euclidean distance since it is proved to approximate the low-dimensional manifold of point set [27]. The smaller the value of k, the better are the results.

2) Discussion about the system parameter: The LCU system has only one parameter: the competition parameter λ. This parameter defines the intensity of competition between particles. When $\lambda = 0$, particles randomly walk the network, without competition. As λ approaches to 1, particles are more likely to compete and, consequently, to be absorbed. Figure 5 depicts the average error of our method with different values of λ. Based on the figure, our model is not sensitive to λ. In general, we suggest setting $\lambda = 1$ because of better and more consistent classification than other values.

3) Discussion about the system iteration stopping parameter: The time limit parameter τ controls when the simulation should stop; it must be at least as large as the diameter of the network. That way, it is guaranteed every edge to be visited by a particle. Since the network diameter is usually a small value, the simulation stops in few iterations.

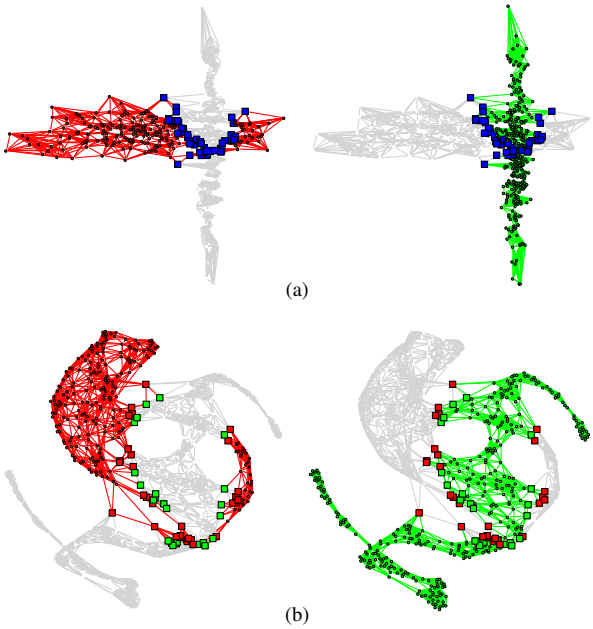


Fig. 6. Unfoldings generated by the proposed system at time $t = 100$ on Highleyman dataset. Edges are colored according to the dominating class at the time. Light gray edges stand for edges presented in the original network but not in the unfolding. (a) Vertex position is imposed by the original data points and blue squares represent vertices connected in both unfoldings. (b) Vertex position is not imposed by the original data points and color of the vertices are the result of the classification.

C. Simulations on Artificial Datasets

For better understanding the details of the LCU system, in this subsection we illustrate it using two synthetic datasets. Each dataset has a different class distribution—banana shape and Highleyman. (The datasets are generated using the PRTools framework [28].) The banana shape dataset is uniformly distributed along specific shape, and then superimposed on a normal distribution with the standard deviation of 1 along the axes. In Highleyman distribution, two classes are defined by bivariate normal distributions with different parameters. Because the datasets are not in a network representation, we use k-NN graph construction method to transform them into respective network form. In the constructed network, a vertex represents a datum, and it connects to its k nearest neighbors, determined by the Euclidean distance. We set $\lambda = 1$ for the simulation.

Firstly, the technique is tested on the Highleyman dataset. Each class has 300 samples, of which 6 are labeled. (We set $k = 10$ for the k-NN algorithm.) We can observe that the labeled data points of the green class form a barrier to samples of the red class. The unfoldings $G^{\text{red}}(100)$ and $G^{\text{green}}(100)$ are presented in Figure 6a. In this figure, blue squares represent vertices that are connected by edges of both unfoldings. Besides of the labeled data of green class forming a barrier, the constructed subnetworks are still connected—there is a single component connecting all the vertices of the subnetwork. It is better visualized in Figure 6b. In this figure, the same unfoldings are presented, but the positions of the vertices are not imposed by the original data. Furthermore, colors of vertices in the figure indicate the result of classification.

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

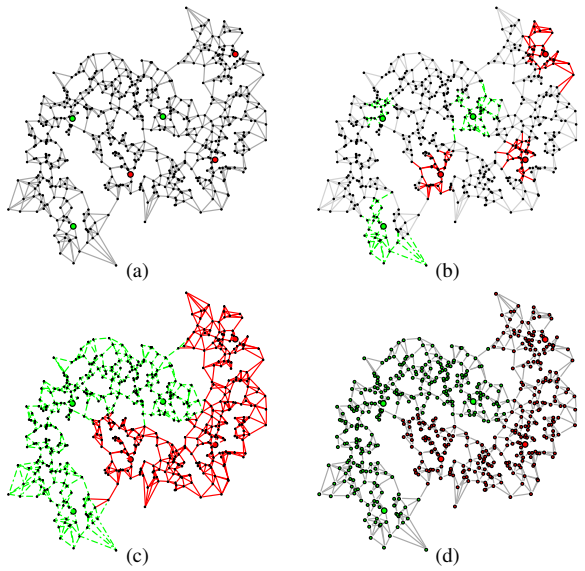


Fig. 7. System evolution on a banana-shaped distribution dataset. Red and green colors represent the two classes. Unlabeled points are black ones; labeled vertices are represented by larger and colored points. Edges are colored according to the dominating class at current iteration, where a light gray point stands for a vertex, which is not dominated yet. (a) The network representation of the dataset at the beginning of the system. (b) and (c) System iteration at time 4 and 20, respectively. (d) The result of the dataset classification.

The overlapping data can be identified by the vertices that belong to two or more unfoldings. This result reveals that the competition system in edges provide more information than the competition in vertices since it can identify the overlapping vertices as part of the system, that is, without special treatments or modifications.

The last synthetic dataset has 600 samples equally split into two classes. In Figure 7a, the initial state of the system is illustrated, where the dataset is represented by a network. (The network representation is obtained by setting $k = 4$ for the k-NN–graph construction.) At this stage, the edges are not dominated by any of the classes. Starting from this state, labeled vertices (sources) generate particles that carry the label of the sources. Though the particles are not shown, Figures 7b and 7c are snapshots of the system evolution—at time 4 and 20—where each edge is colored by its dominating class at that iteration. In these illustrations, a solid red line stands for an edge (i, j) that $\delta_{ij}^{\text{red}} + \delta_{ji}^{\text{red}} > \delta_{ij}^{\text{green}} + \delta_{ji}^{\text{green}}$, while a dashed green line stands for the opposite. When $\delta_{ij}^{\text{red}} + \delta_{ji}^{\text{red}} = \delta_{ij}^{\text{green}} + \delta_{ji}^{\text{green}}$, an edge is drawn in a solid light gray line. As expected, edges close to sources are dominated initially, and farther edges are progressively dominated. At time 20, Figure 7c, every edge has been dominated, and the edge domination does not change anymore. Figure 7d shows dataset classification following the system result. In this example, with 1% of points in the labeled set, the technique can correctly identify the pattern formed by each class. Both results are satisfactory, reinforcing the ability of the technique of learning arbitrary class distributions.

TABLE II
TEST ERRORS (%) WITH STANDARD DEVIATION AND THE BEST PARAMETERS

	10 labeled	k	λ	**100 labeled**	k	λ
g241c	42.90 ± 4.33	10	0.25	30.03 ± 2.18	10	0.875
g241n	46.94 ± 3.93	9	0	36.08 ± 6.32	9	0
Digit1	4.93 ± 2.63	5	0.75	1.51 ± 0.31	6	0.625
USPS	15.65 ± 3.81	3	1	8.36 ± 2.92	3	1
COIL	59.96 ± 6.13	3	0.625	13.73 ± 2.91	3	0
BCI	47.56 ± 1.80	9	1	34.68 ± 2.26	3	0.25
Text	29.71 ± 3.53	9	0.875	22.41 ± 1.74	10	0.75

TABLE III
TEST ERRORS (%) WITH 10 LABELED TRAINING POINTS

	g241c	**g241d**	**Digit1**	**USPS**	**COIL**	**BCI**	**Text**	**Avg. Rank**
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12	9.3
SVM	47.32	46.66	30.60	20.03	68.86	49.85	45.37	13.0
MVU + 1-NN	47.15	45.56	14.42	23.34	62.62	47.95	45.32	9.3
LEM + 1-NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44	9.1
QC + CMR	39.96	46.55	9.80	13.61	59.63	50.36	40.79	6.9
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37	10.4
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21	10.0
Cluster–Kernel	48.28	42.05	18.73	19.41	67.32	48.31	42.72	10.1
LDS	28.85	50.63	15.63	17.57	61.90	49.27	27.15	8.0
Laplacian RLS	43.85	45.68	5.44	18.99	54.54	48.97	33.68	5.9
LGC	45.82	44.09	9.89	9.03	63.45	47.09	46.83	6.9
LP	42.61	41.93	11.31	14.83	55.82	46.37	49.53	5.1
LNP	47.82	46.24	8.58	17.87	55.50	47.65	41.06	7.1
Original Particle Competition	41.17	43.51	8.10	15.69	54.18	48.00	34.84	4.0
Labeled Component Unfolding	42.90	46.94	4.93	15.65	59.96	47.56	29.71	4.9

TABLE IV
TEST ERRORS (%) WITH 100 LABELED TRAINING POINTS

	g241c	**g241d**	**Digit1**	**USPS**	**COIL**	**BCI**	**Text**	**Avg. Rank**
1-NN	43.93	42.45	3.89	5.81	17.35	48.67	30.11	11.4
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45	8.1
MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89	32.83	10.6
LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77	10.9
QC + CMR	22.05	28.20	3.15	6.36	10.03	46.22	25.71	6.6
Discrete Reg.	43.65	41.65	2.77	4.68	9.61	47.67	24.00	7.1
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52	7.7
Cluster-Kernel	13.49	4.95	3.79	9.68	21.99	35.17	34.28	7.4
LDS	18.04	23.74	3.46	4.96	13.72	43.97	23.15	5.4
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	31.36	23.57	4.4
LGC	41.64	40.08	2.72	3.68	45.55	43.50	56.83	9.3
LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79	8.3
LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.45	11.4
Original Particle Competition	21.41	25.85	3.11	4.82	10.94	41.57	27.92	5.3
Labeled Component Unfolding	30.03	36.08	1.51	8.36	13.73	34.68	22.41	6.0

D. Simulations on Benchmark Datasets

We compare our model with 14 semi-supervised techniques tested on Chapelle's benchmark [1]. The benchmark is formed by seven datasets that have 1500 data points, except for BCI that has 400 points. The datasets are described in [1].

For each dataset, 24 distinct, unbiased sets (splits) of labeled points are provided within the benchmark. Half of the splits are formed by 10 labeled points and the other half by 100 labeled points. The author of the benchmark ensured that each split contains at least one data point of each class. The result is the average test error—the proportion of data points incorrectly labeled—over the splits. We compare our results to the ones obtained by the following techniques: 1-Nearest Neighbors (1-NN), Support Vector Machines (SVM), Maximum variance unfolding (MVU + 1-NN), Laplacian eigenmaps (LEM + 1-NN), Quadratic criterion and class mass regularization (QC + CMR), Discrete regularization (Discrete reg.), Transductive support vector machines (TSVM), Cluster kernels (Cluster-Kernel), Low-density separation (LDS), Laplacian regularized least squares (Laplacian RLS), Local and global consistency (LGC), Label propagation (LP), Linear neighborhood propa-

gation (LNP), and Network-Based Stochastic Semisupervised Learning (Vertex Domination), The simulation results are collected from [1], except for LGC, LP, LNP, and Original Particle Competition that are found in [17].

For the simulation of the LCU system, we discretize the interval of the parameter in $\lambda = \{0, 0.125, \ldots, 1\}$. Also, we vary the k-NN parameter $k \in \{1, 2, \ldots, 10\}$. We tested every combination of k and λ. Moreover, we fix $\tau = 1000$. In Table II, we present the results with the standard deviation over the splits along with the best combination of parameters that generated the best accuracy result.

The test error comparison for 10 labeled points are shown in Table III; comparison for 100 labeled points are in Table IV. Apart from each dataset, the last column is the average performance rank of a technique over the datasets. A ranking arranges the methods under comparison by test error rate in ascending order. For a single dataset, we assign rank 1 for the method with the lowest average test error on that dataset, then rank 2 for the method with the second lowest test error, and so on. The average ranking is the average value of the rankings of the method on all the datasets. The smaller the ranking score, the better the method has performed.

From the average rank column, the LCU technique is not the best ranked, but it is in the best group of techniques in both 10 labeled and 100 labeled cases.

We statistically compare the results presented in Tables III and IV. For all tests we set a significance level of 5%. First, we use a test based on the average rank of each method to evaluate the null hypothesis that all the techniques are equivalent. With the Friedman test [29], there is statistically significant difference between the rank of the techniques

Since the Friedman test result reports statistical significance, we use the Wilcoxon signed-rank test [29]. In this pairwise difference test, we test for the null hypothesis that the first technique has greater or equal error results than the second. If rejected at a 5% significance level, then we say the first technique is superior to the second. By analyzing results for 10 and 100 labeled points together, we conclude that our technique is superior to 1-NN, LEM + 1-NN, and MVU + 1-NN. Examining separately, for 10 labeled points, our method is also superior to discrete regularization, cluster kernel, and SVM. For 100 labeled points, it is also superior to LNP and LGC; whereas Laplacian RLS is superior to ours.

E. Simulations on Human Activity Dataset

The Human Activity Recognition Using Smartphones [30] dataset comprises of 10299 data samples. Each sample matches 561 features extracted from motion sensors attached to a person during a time window. Each person performed six activities which are target labels in the dataset—walking (WK), walking upstairs (WU), walking downstairs (WD), sitting (ST), standing (SD), and laying down (LD).

We use k-NN with $k = 7$ for the dataset network representation once it is the smallest value that generates a connected network. The parameters are fixed in $\lambda = 1$ and $\tau = 1000$. We compare our results with the ones published in [30], splitting the problem into six binary classification tasks.

Table V summarises the results. For our technique, we provide the precision, recall and F Score using 5%, 10%, and 20% of labeled samples. We average the results of 10 independent labeled set for each configuration. We also provide the original results from [30] using SVM with approximately 70% of labeled samples. Our technique performs as well as SVM *using far fewer labeled samples* and using the suggested parameter set. Such a feature is quite attractive because it may represent a big saving in money or efforts when involving manually data labeling in semi-supervised learning.

F. Simulations on MNIST Dataset

The MNIST dataset comprises 70,000 examples of handwritten digits. All digits have been size-normalized and centered in a fixed-size image. In a supervised learning setting, this dataset is split into two sets: 60,000 examples for training and 10,000 for testing.

To adapt the dataset to a semi-supervised learning problem, we use a setting similar to [23], [31]: the labeled input data items are selected from the training set, and the unlabeled ones from the test set. Although we do not use a validation set, [23] and [31] use an additional set of at least 1,000 labeled samples for parameter tuning.

The network representation is obtained from the images without preprocessing. We use the Euclidean distance between items and $k = 3$ to construct the k-NN network. Similarly to the previous experiment, the value of k is the smallest value that generates a connected network. The parameters are fixed in $\lambda = 0.9$ and $\tau = 500$.

Table VI compares the error rate of our method to other semi-supervised techniques. To the best of our knowledge, we could not find many papers with experiments under the same semi-supervised settings for the MNIST dataset. Due to such available results in the literature, we sought to carry our experiments with the input as similar as possible to the compared results. However, a single sampling with as less as 1,000 labeled samples out of a set of 60,000 images most probably results in a biased accuracy result, we opt to average results from 15 labeled sets for each parameter setting.

Our model performs well even without preprocessing and validation set. This result indicates that besides the simplicity of the constructed network, our learning system can obtain enough knowledge from data.

V. Conclusion

We have presented a transductive semi-supervised learning technique based on a vertex-edge dynamical system on complex networks. First, the input data is mapped into a network. Then, the proposed Labeled Component Unfolding (LCU) system runs on this network. At this stage, particles compete for edges in the network. When a particle passes through an edge, it increases its class dominance over the edge while decreasing other classes' dominance. Three dynamics—walking, absorption and production—provide a biologically inspired scenario of competition and cooperation. Then, labels are assigned according to the dominant class over the edges. As a result, the system unfolds the original network by

ACCEPTED VERSION - DOI 10.1109/TNNLS.2016.2626341

TABLE V
PERFORMANCE COMPARISON IN THE HUMAN ACTIVITY RECOGNITION USING SMARTPHONES DATASET

	Labeled Component Unfolding									SVM		
	5% labeled			10% labeled			20% labeled			≈70% labeled [30]		
	Precision	Recall	F Score	Precision	Recall	F Score	Precision	Recall	F Score	Precision	Recall	F Score
WK	.984 ± .013	.941 ± .030	.962 ± .016	.992 ± .004	.985 ± .011	.989 ± .006	.994 ± .002	.997 ± .001	**.995 ± .001**	.957	.992	.974
WU	.981 ± .009	.935 ± .026	.957 ± .015	.988 ± .008	.961 ± .013	.974 ± .008	.991 ± .003	.981 ± .006	**.986 ± .004**	.980	.958	.969
WD	.987 ± .017	.901 ± .016	.942 ± .011	.994 ± .008	.918 ± .011	.955 ± .007	.998 ± .001	.945 ± .008	.971 ± .004	.988	.976	**.982**
ST	.864 ± .034	.698 ± .049	.770 ± .022	.883 ± .015	.743 ± .039	.806 ± .020	.905 ± .014	.814 ± .015	.857 ± .006	.969	.880	**.922**
SD	.840 ± .024	.842 ± .053	.839 ± .017	.870 ± .023	.844 ± .022	.856 ± .006	.896 ± .013	.872 ± .021	.884 ± .009	.901	.974	**.936**
LD	.996 ± .002	.999 ± .000	.998 ± .001	.997 ± .001	.999 ± .000	.998 ± .000	.997 ± .001	.999 ± .000	.998 ± .000	1.000	1.000	**1.000**

TABLE VI
TEST ERRORS (%) IN THE MNIST DATASET

Method	100 labeled	1000 labeled
LCU	10.62 ± 1.91	6.31 ± 0.46
TSVM* [23]	16.81	5.65
Embed NN* [31]	16.86	8.52
Embed CNN* [31]	7.75	3.82

* The comparison is biased since the results from [23], [31] rely on a single and unique labeled set. See text for more details.

grouping edges dominated by the same class. Finally, we employ the unfoldings to classify unlabeled data. Furthermore, rigorous studies have been done on the novel LCU system.

The deterministic system implementation brings advantages over its stochastic counterpart. The time complexity of the deterministic one does not depend on the number of particles, so we are benefited from better results when considering a continuously varying number of initial particles. Besides, the LCU system allows a stable transductive semi-supervised learning technique with a subquadratic order of complexity. Computer simulations show the proposed technique achieves a good classification accuracy and it is suitable for situations where a small number of labeled samples are available. Another interesting feature of the proposed model is that it directly provides the overlapping information of each vertex or a subset of vertices.

As future works, we would like to investigate the mathematical property of the LCU system on directed or weighted networks. Besides of this, it is interesting to improve the runtime further via network sampling methods or estimation methods. In this way, the model will be suitable to be applied to process large enough datasets or streaming data. Another interesting research is to treat the labels on edges instead of nodes in a semi-supervised learning environment.

REFERENCES

[1] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[2] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, 3rd ed. Morgan and Claypool Publishers, 2009.

[3] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[4] M. Loog and A. C. Jensen, "Semi-Supervised Nearest Mean Classification Through a Constrained Log-Likelihood," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 995–1006, 2015.

[5] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proc. of the 18th International Conference on Machine Learning*, 2001, pp. 577–584.

[6] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, 2005.

[7] V. N. Vapnik, *Statitical learning theory*. New York, NY: Wiley, 1998.

[8] T. C. Silva and L. Zhao, "Semi-supervised learning guided by the modularity measure in complex networks," *Neurocomputing*, vol. 78, no. 1, pp. 30–37, 2012.

[9] L. Cheng and S. J. Pan, "Semi-supervised Domain Adaptation on Manifolds," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2240–2249, 2014.

[10] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling Up Graph-Based Semisupervised Learning via Prototype Vector Machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 444–457, 2015.

[11] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.

[12] M. Newman, A.-L. Barabási, and D. J. Watts, *The Structure and Dynamics of Networks: Princeton Studies in Complexity*. Princeton University Press, 2006.

[13] M. E. J. Newman, *Networks: An Introduction*, 1st ed. New York, NY: Oxford University Press, 2010.

[14] T. C. Silva and L. Zhao, *Machine Learning in Complex Networks*, 1st ed. Springer, 2016.

[15] X. Zhu, A. B. Goldberg, and T. Khot, "Some new directions in graph-based semi-supervised learning," *Proc. of the EEE International Conference on Multimedia and Expo*, pp. 1504–1507, 2009.

[16] M. G. Quiles, L. Zhao, R. L. Alonso, and R. A. F. Romero, "Particle competition for complex network community detection," *Chaos*, vol. 18, no. 3, p. 033107, 2008.

[17] T. C. Silva and L. Zhao, "Network-based stochastic semisupervised learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 451–66, 2012.

[18] ——, "Network-based stochastic semisupervised learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 451–466, 2012.

[19] ——, "Detecting and preventing error propagation via competitive learning," *Neural Networks*, vol. 41, pp. 70–84, 2013.

[20] J. Jensen, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," *Acta Mathematica*, vol. 30, no. 1, pp. 175–193, 1906.

[21] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, 2004, pp. 321–328.

[23] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Large Scale Transductive SVMs," *Journal of Machine Learning Research*, vol. 7, no. 7, pp. 1687–1712, 2006.

[24] B. Wang, Z. Tu, and J. K. Tsotsos, "Dynamic label propagation for semi-supervised multi-class multi-label classification," *Proc. of the IEEE International Conference on Computer Vision*, pp. 425–432, 2013.

[25] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *School Comput Sci Carnegie Mellon Univ Pittsburgh PA Tech Rep*, vol. 54, no. CMU-CALD-02-107, pp. 1–19, 2002.

[26] Y. M. Zhang, K. Huang, G. G. Geng, and C. L. Liu, "MTC: A Fast and Robust Graph-Based Transductive Learning Method," *IEEE Trans Neural Netw Learn Syst*, vol. 26, no. 9, pp. 1979–1991, 2014.

[27] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, p. 2319, 2000.

[28] R. Duin, P. Juszczak, P. Paclík, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "PR-Tools4.1, a matlab toolbox for pattern recognition," 2007.
[29] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*, 2nd ed. Wiley-Interscience, 1999.
[30] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," in *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, 2013.
[31] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *In Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.

CHAPTER

3

ADVANTAGES OF EDGE-CENTRIC COLLECTIVE DYNAMICS IN MACHINE LEARNING TASKS

© 2018 L&H Scientific Publishing, LLC. All rights reserved. No redistribution is permitted. Accepted version of Filipe A. N. Verri, Paulo R. Urio, Liang Zhao, “Advantages of Edge-Centric Collective Dynamics in Machine Learning Tasks”, Journal of Applied Nonlinear Dynamics, Vol. 7(3), September, 2018.

Contribution statement

F.A.N. Verri conceived and planned the experiments. F.A.N. Verri proposed the artificial network model and the vertex-centric system. F.A.N. Verri devised the analytical results. P.R. Urio carried out the experimental simulations and produced the resulting figures. P.R. Urio, F.A.N. Verri, and L. Zhao contributed to the writing of the manuscript. L. Zhao supervised the project.

Journal of Applied Nonlinear Dynamics 7(3) (2018) 1-17

Journal of Applied Nonlinear Dynamics

https://lhscientificpublishing.com/Journals/JAND-Default.aspx

Advantages of Edge-Centric Collective Dynamics in Machine Learning Tasks

Filipe Alves Neto Verri[1†], Paulo Roberto Urio[2], and Liang Zhao[3]

[1]Institute of Mathematical and Computer Sciences, University of São Paulo, São Carlos, Brazil
School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe AZ, USA
[2]Institute of Mathematical and Computer Sciences, University of São Paulo, São Carlos, Brazil
[3]Ribeirão Preto School of Philosophy, Science and Literature, University of São Paulo, Ribeirão Preto, Brazil

Submission Info

Estimated dates by the authors
Received 31 Aug 2016
Accepted 12 Apr 2017
Available online 01 Sep 2018

Keywords

Complex networks
Nonlinear dynamics
Collective dynamics
Particle competition
Machine learning

Abstract

We study how effectively edge-centric dynamics solve semi-supervised learning tasks. The Edge Domination System is an algorithm to reveal patterns and obtain information of the underlying complex network. The algorithm consists of the simulation of a collective dynamical system based on particle competition for the dominance of edges. In this paper, we propose a vertex-centric version of this model and assess the differences between the edge-centric model. The edge-centric system offers better features in semi-supervised learning tasks, such as greater exploration behavior and faster convergence.

©2016-2018 L&H Scientific Publishing, LLC. All rights reserved.
(Accepted version.)

1 Introduction

We study emergent collective behavior in dynamical systems and how it can benefit an algorithm in data learning tasks. The study of systems with the emergence of collective dynamics has led to algorithms based on the behavior of birds, bees, ants, bacterial colony, and many other forms of intelligence. In these biological situations, the driving factor of interactions ultimately aims at the individuals' survival, which manifests in the behavior of cooperation, competition, or a compound of them. Such behaviors often inspire machinery for solutions in optimization and learning problems and are increasingly found in the literature [1–3].

We introduced in [4] the Edge Domination System (EDS), an algorithm that deterministically obtains information of the underlying complex network by exploring patterns revealed by the collective dynamics of particle competition. The algorithm models particles that randomly walk a complex network, represented by a graph, in which teams of particles dominate the edges they visited more frequently than their rivals. Afterward, the algorithm promotes an *unfolding*: it breaks the network

†Corresponding author.
Email address: filipeneto@usp.br

ISSN 2164-6457, eISSN 2164-6473/$-see front materials © 2018 L&H Scientific Publishing, LLC. All rights reserved.
This is an accepted version.

down in subnetworks induced by the edges and their dominant label. The unfolded networks allow the algorithm to draw final conclusions regarding the labels of the vertices, which is the result of the learning task.

Unlike similar approaches with particle competition [5], the edges are the resource the particles compete for instead of the vertices. Since this approach is rather new, the advantages of edge competition are still unclear and need to be explored. We presented an initial assessment in [6], and in this paper, we will further study the differences between the two resources of competition. We will model a comparable algorithm of particles competing for vertices, and we will compare it with EDS, whose particles compete for edges.

We conclude that the edge-centric dynamics is preferred over the vertex-centric, since it acquires more information and converges faster at the same computational cost.

The remaining of this paper is organized as follows. Section 2 reviews the original EDS and introduces an artificial network model to represent learning problems. The effectiveness of a simplified version of EDS and its vertex-based counterpart are compared in Section 3. The convergence of the classification results in both systems is studied in Section 4. Section 5 compares the systems in benchmark datasets according to convergence and classification error. Finally, Section 6 concludes the paper.

2 Study framework

In this section, we review the original Edge Domination System and propose an artificial network model. We use this network model to represent the learning scenarios with controllable difficulty, in which we compare EDS and its vertex-based counterpart.

2.1 Review of Edge Domination System

The Edge Domination System [4] models a process in a complex network in which teams of particles compete to dominate the largest number of edges. The interaction network is represented by a graph $G = (\mathscr{V}, \mathscr{E})$ that is simple, unweighted, and undirected. The set of vertices $\mathscr{V} = \mathscr{L} \cup \mathscr{U}$ splits vertices in l labeled vertices, $\mathscr{L} = \{v_1, \ldots, v_l\}$, and u unlabeled vertices, $\mathscr{U} = \{v_{l+1}, \ldots, v_{l+u}\}$, with a total of $|\mathscr{V}| = l + u$ vertices, where we suppose few labeled vertices, $l \ll u$. The problem contains $C > 1$ possible labels, and each v_i from the labeled set $\mathscr{L}$ has the label $y_i \in \{1, \ldots, C\}$. The set of edges $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ contains an edge (i, j) if v_i and v_j are connected. The network is also represented by the adjacency matrix $A = (a_{ij})$ where $a_{ij} = a_{ji} = 1$ if the edge (i, j) exists.

One team of particles exists for each possible label, totalizing C teams of particles and vertices. All the particles randomly walk from vertex to vertex, one vertex at a time. When any particle passes through an edge, the edge counts the visit for the particle's team. An edge registers two values for a team: the number of visits solely from the current time, and the cumulative number of visits since the beginning. At any moment, the team that dominates an edge is determined by the team with the highest number of visits. The system models no cooperation between teams, and any two particles or vertices are rival if they belong to different teams. Particles are always bonded to the same team, but the size of the teams vary because particles can be removed from the system, or new particles can be generated by the labeled vertices. These rules are described below as three actions, which combine to model the evolution rule of the system.

Walking. A particle is always in a vertex and chooses a connected vertex to move at each time, with equal probability between the neighbors of the current vertex.

Absorption. The transitory *team domination* is the number of visits in an edge by teams that occurred at the last time. In the next movement, the transitory team domination determines the success rate of particles moving to their next vertices, or to fail and being removed from the system—the higher

the presence of a team, the easier particles from the same team pass through the edge. Additionally, all the labeled vertices thwart every rival particle trying to reach them, automatically removing those particles from the system—this action characterizes such vertices as *sinks*.

Generation. Besides being sinks, the labeled vertices are also *sources* for new particles. A source vertex generates particles for its team according to the neighborhood size and the size of the population. This rule maintains the size of the population close to its initial size.

Every edge has transitory and cumulative values of team domination, which assist an algorithm solving machine learning problems. An approach considers only the team that dominates each edge, grouping the edges by the team that dominates. The algorithm unfolds the interaction network in subnetworks by group: the *unfolding* of team c at time t is the subnetwork containing all the edges dominated by team c at time t. Over the evolution of the competition process, the network's unfoldings change less and less until they converge to stable subnetworks that are afterward analyzed to the machine learning task.

In [4], we proposed a system that describes the expected evolution of the particles over time. This system models deterministic evolution rules, and it is denoted as

$$X(t) = \begin{bmatrix} \mathbf{n}^c(t) = \left[n_i^c(t)\right]_{i=1,\dots,|\mathscr{V}|} \\ N^c(t) = \left(n_{ij}^c(t)\right)_{i,j=1,\dots,|\mathscr{V}|} \\ \Delta^c(t) = \left(\delta_{ij}^c(t)\right)_{i,j=1,\dots,|\mathscr{V}|} \end{bmatrix}_{c=1,\dots,C}, \tag{1}$$

where $\mathbf{n}^c(t)$ is a row vector whose elements $n_i^c(t)$ describe the population of particles from team c in each vertex v_i at time t. Stored in sparse matrices, the elements $n_{ij}^c(t) \in N^c(t)$ and $\delta_{ij}^c(t) \in \Delta^c(t)$ are the *transitory directed domination* and the *cumulative domination*, in that order. At time t, $n_{ij}^c(t)$ is the portion of particles from team c that moved from v_i to v_j, while $\delta_{ij}^c(t)$ is the accumulated portion of visits since the beginning.

The system X is a nonlinear Markovian dynamical system with the deterministic evolution function

$$\phi\colon \begin{cases} \mathbf{n}^c(t+1) = \mathbf{n}^c(t) \times P^c(X(t)) + \mathbf{g}^c(X(t)) \\ N^c(t+1) = \operatorname{diag} \mathbf{n}^c(t) \times P^c(X(t)) \\ \Delta^c(t+1) = \Delta^c(t) + N^c(t+1) \end{cases}, \tag{2}$$

where $\operatorname{diag}\mathbf{v}$ is a square matrix with the elements of vector $\mathbf{v}$ on the main diagonal and $\times$ stands for the vector–matrix product.

The function $P^c(X(t))$ of the system X at time t is a square matrix whose elements are

$$p_{ij}^c(X(t)) = \begin{cases} 0 & \text{if } v_j \in \mathscr{L} \text{ and } y_j \neq c\,, \\ \dfrac{a_{ij}}{\deg v_i}\left(1 - \lambda\, \sigma_{ij}^c(X(t))\right) & \text{otherwise,} \end{cases} \tag{3}$$

where

$$\sigma_{ij}^c(X(t)) = \begin{cases} 1 - \dfrac{n_{ij}^c(t) + n_{ji}^c(t)}{\sum_{q=1}^{C} n_{ij}^q(t) + n_{ji}^q(t)} & \text{if } \sum_{q=1}^{C} n_{ij}^q(t) + n_{ji}^q(t) > 0, \\ 1 - \dfrac{1}{C} & \text{otherwise.} \end{cases} \tag{4}$$

Given that we know the initial state $X(0)$ of the system, the function $\mathbf{g}^c(X(t))$ of the system X at time t returns a row vector where the i-th element is

$$g_i^c(X(t)) = \rho_i^c \max\{0,\, \mathbf{1}\cdot\mathbf{n}^c(0) - \mathbf{1}\cdot\mathbf{n}^c(t)\}\,, \tag{5}$$

4

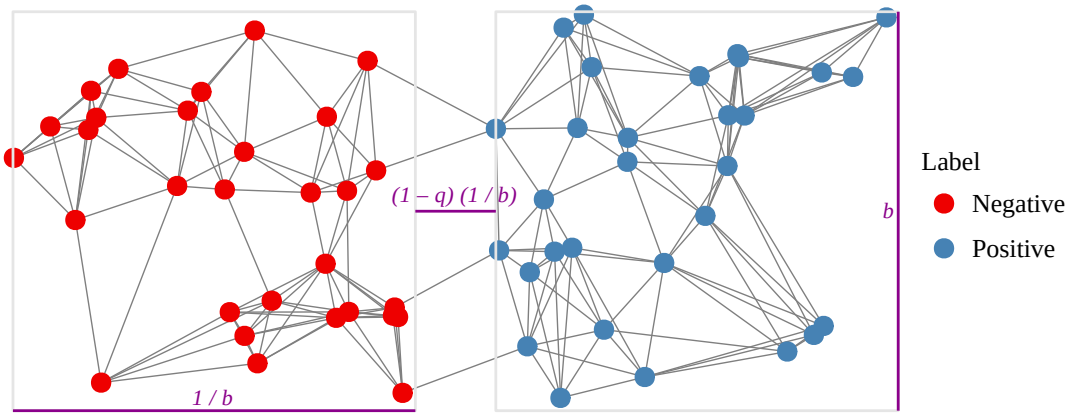


Fig. 1 Example of network $G(60,6,0.8,1)$. Vertices are colored according with their label and positioned according with their associated point in the space.

where $\mathbf{1}$ is a row vector whose elements are 1, '$\cdot$' is the inner product between vectors, and

$$\rho_i^c = \begin{cases} \dfrac{\deg v_i}{\sum_{v_j \in \mathscr{G}^c} \deg v_j} & \text{if } v_i \in \mathscr{G}^c, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Using the cumulative domination, we group edges by team domination. For each team c, the subset of edges $\mathscr{E}^c(t) \subseteq \mathscr{E}$ contains the edges dominated by the team c,

$$\mathscr{E}^c(t) = \left\{ (i,j) \middle| \arg\max_q \left(\delta_{ij}^q(t) + \delta_{ji}^q(t) \right) = c \right\}. \tag{7}$$

We define the subnetwork

$$G^c(t) = (\mathscr{V}, \mathscr{E}^c(t)) \tag{8}$$

as the *unfolding* of network G obtained with the edges of team c at time t.

The initial state of the system X is determined by an arbitrary *population size* $n_i^c(0)$ of initial active particles and

$$\begin{cases} n_{ij}^c(0) = 0, \\ \delta_{ij}^c(0) = 0. \end{cases} \tag{9}$$

2.2 Artificial network model

To study the EDS in semi-supervised learning tasks, we introduce a model of a random network with labeled vertices. The network model specifies a binary classification task with controllable overlap between vertices with different labels.

The undirected random network $G(n,k,q,b)$, as shown in Fig. 1, contains n vertices of which half of them are labeled as positive. The other half is labeled as negative. Each vertex v_i is associated with a point x_i in a bi-dimensional space. Let $y_i \in \{+,-\}$ be the label of vertex v_i, the points x_i such that $y_i = -$ is uniformly distributed in the interval $\left[0, b^{-1}\right] \times [0,b]$, and the points x_j such that $y_j = +$ is in the interval $\left[(2-q)b^{-1}, (3-q)b^{-1}\right] \times [0,b]$. The two vertices v_i and v_j are connected if x_i is within the k-neighborhood of x_j, according to the Euclidean metric, or vice-versa.

The parameter $q \in [0,2]$ controls the overlapping area between the surfaces from which the points are sampled. If $q < 1$, the surfaces do not overlap, and the inter-label region of the resulting network is less dense than the intra-label. If $q = 1$, the density of the resulting graph is uniform along the graph.

5

If $q > 1$, an overlapping region exists with a high density of edges. In the extreme $q = 2$, points with different labels are sampled from the same surface. The parameter $b > 0$ controls the size of the border, which increases the number of vertices that may have at least one connection with vertices with the other label.

The networks $G(n,k,q,b)$ are, thus, spatial networks with controllable border and overlap between vertices with different labels. The *border size* is b and the *overlapping area* is $\max(0, q-1)$.

3 Effectiveness in different levels of difficulty

We look for differences and similarities between the edge-based competition system and the vertex-based competition system. Since no study of the differences has been carried out before, we first simplify the previously proposed model.

We slightly modify the original EDS under certain assumptions that should not impair the performance in machine learning problems, and compare both models in the synthetic networks that provide controllable learning scenarios (see Section 2.2).

3.1 Removal of cumulative domination

We assume two properties: the system's unfoldings always converge, and the initial population is balanced between vertices with different labels.

The assumption of unfolding convergence. With this assumption, we remove the cumulative domination from the model and modify the unfolding strategy to use the transitory domination. In the original model, the cumulative domination determined the unfoldings of the network. But, we observed that after few iterations the transitory domination becomes proportional to the cumulative domination, which allowed us to simplify the unfolding process. In fact, if the matrices $N^c(t)$ converge for all c,

$$\lim_{t\to\infty} \arg\max_c \left(\delta_{ij}^c(t) + \delta_{ji}^c(t)\right) = \lim_{t\to\infty} \arg\max_c \left(n_{ij}^c(t) + n_{ji}^c(t)\right). \tag{10}$$

The same limiting unfoldings can be obtained without the accumulated domination. The convergence of the unfoldings are analytically and experimentally studied in Section 4.

Moreover, we have verified that the unfoldings are convergent in all simulations carried out in this paper.

The assumption of equality of the initial population. In [4], the semi-supervised learning method based on the EDS uses an initial population of particles distributed according to the degrees of the vertices, without considering the team distribution. Conversely, we adopt a less restrictive assumption of the initial condition. We assume there is no preferred label, that is, given $n_0 > 0$,

$$\sum_i n_i^c(0) = n_0, \tag{11}$$

for all labels c.

With this assumption, and relying on the scale-invariant property of the system, we simplify the particle generation term shown in Eq. (5). The scale-invariant property means that multiplying the initial population of particles in every vertex by a positive constant will scale the population at any time by the same factor. Because we need to know which team dominates and not the size of each team, we may scale any initial population to be a discrete distribution of particles while maintaining the same resulting unfoldings—we multiply the initial population of each vertex by n_0^{-1}. Since the sum

of a distribution is always 1, the size of a team's population is also 1 at the beginning.

Under such assumptions, we prove the size of every team never exceeds 1. The population limit enables us to drop the max function in Eq. (5).

Theorem 1. *Population limit. The population's size $\sum_i n_i^c(t)$ of each label c is smaller than or equal to 1 if the initial population is $\sum_i n_i^c(0) = 1$.*

Proof. We know that

$$n_i^c(1) = \sum_j n_j^c(0) p_{ji}^c(X(0)) \leq \sum_j n_j^c(0) \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \frac{C-1}{C}\right)$$

and thus

$$\sum_i n_i^c(1) \leq \left(1 - \lambda \frac{C-1}{C}\right) \sum_j n_j^c(0) \sum_i \frac{a_{ji}}{\deg v_j} \leq \sum_j n_j^c(0) = 1.$$

Now, we assume $\sum_i n_i^c(t) \leq 1$ for all c and $t > 1$, and prove the inequality for $t+1$.

$$\begin{aligned} n_i^c(t+1) = \sum_j n_j^c(t) p_{ji}^c(X(t)) + g_i^c(X(t)) \leq \\ \sum_j n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \sigma_{ji}^c(X(t))\right) + \rho_i^c \max\{0,\, \mathbf{1} \cdot \mathbf{n}^c(0) - \mathbf{1} \cdot \mathbf{n}^c(t)\} \leq \\ \sum_j n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \sigma_{ji}^c(X(t))\right) + \rho_i^c \max\{0,\, 1 - \mathbf{1} \cdot \mathbf{n}^c(t)\} \leq \\ \sum_j n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \sigma_{ji}^c(X(t))\right) + \rho_i^c \left(1 - \sum_k n_k^c(t)\right), \end{aligned} \tag{12}$$

and thus

$$\begin{aligned} \sum_i n_i^c(t+1) \leq \sum_i \sum_j n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \sigma_{ji}^c(X(t))\right) + \left(1 - \sum_k n_k^c(t)\right) \sum_i \rho_i^c \leq \\ \sum_j n_j^c(t) \sum_i \frac{a_{ji}}{\deg v_j} \left(1 - \lambda \sigma_{ji}^c(X(t))\right) + 1 - \sum_k n_k^c(t) \leq \sum_j n_j^c(t) + 1 - \sum_k n_k^c(t) \leq 1. \end{aligned} \tag{13}$$

By mathematical induction, Theorem 1 is proved for all time t and team c.

3.1.1 Edge Domination System

From the assumptions, the simplified system based on edges is

$$X(t) = \begin{bmatrix} \mathbf{n}^c(t) = \left[n_i^c(t)\right]_{i=1,\ldots,|\mathcal{V}|} \\ N^c(t) = \left(n_{ij}^c(t)\right)_{i,j=1,\ldots,|\mathcal{V}|} \end{bmatrix}_{c=1,\ldots,C}, \tag{14}$$

with the evolution

$$\phi: \begin{cases} \mathbf{n}^c(t+1) = \mathbf{n}^c(t) \times P^c(X(t)) + \mathbf{g}^c(X(t)) \\ N^c(t+1) = \operatorname{diag} \mathbf{n}^c(t) \times P^c(X(t)), \end{cases} \tag{15}$$

such that $P^c(X(t))$ is Eq. (3) and $\mathbf{g}^c(X(t))$ is a row vector where the i-th element is

$$g_i^c(X(t)) = \rho_i^c \left(1 - \mathbf{1} \cdot \mathbf{n}^c(t)\right). \tag{16}$$

7

The initial state is given by an arbitrary *discrete distribution* $\mathbf{n}^c(0)$ and the zero matrix $N^c(0)$. The unfoldings are hereafter based on the transitory domination instead of the cumulative. Thus, the unfoldings are $G^c(t) = (\mathscr{V}, \mathscr{E}^c(t))$ such that

$$\mathscr{E}^c(t) = \left\{ (i,j) \middle| \arg\max_q \left(n_{ij}^q(t) + n_{ji}^q(t) \right) = c \right\}. \tag{17}$$

3.1.2 Vertex Domination System

From system in Eq. (14), it is straightforward to shift the competition to the vertices. We eliminate the states that track the transitory domination in the edges and rewrite σ, Eq. (4), in terms of the transitory domination in the vertices. The Vertex Domination System (VDS) is

$$X(t) = \left[\mathbf{n}^c(t) = \left[n_i^c(t) \right]_{i=1,\dots,|\mathscr{V}|} \right]_{c=1,\dots,C}, \tag{18}$$

with the evolution function

$$\mathbf{n}^c(t+1) = \mathbf{n}^c(t) \times P^c(X(t)) + \mathbf{g}^c(X(t)), \tag{19}$$

where $P^c(X(t))$ is Eq. (3) with

$$\sigma_{ij}^c(X(t)) = \begin{cases} 1 - \dfrac{n_j^c(t)}{\sum_{q=1}^C n_j^q(t)} & \text{if } \sum_{q=1}^C n_j^q > 0, \\ 1 - \dfrac{1}{C} & \text{otherwise,} \end{cases} \tag{20}$$

and $\mathbf{g}^c(X(t))$ is a row vector where the i-th element is Eq. (16).

The initial state of the system X is given by an arbitrary *discrete distribution* $\mathbf{n}^c(0)$. The concept of unfoldings is not defined in this system.

3.2 Experimental simulations

We formulate a simple example where Edge Domination System (EDS) and Vertex Domination System (VDS) behave differently. With this example, we reason the differences and the impact of such differences on semi-supervised learning tasks. Afterward, we compare the effectiveness of both systems in artificial data.

3.2.1 Toy example

In Fig. 2(a), the vertices can be either *positive* or *negative*, and we observe the label found for the central vertex, which has no label previously defined. With this network, one shall expect the fraction of positive particles in the central vertex to be the same for EDS and VDS. As it indeed happens in Fig. 3(a), both systems end with the same fraction of particles in every vertex.

However, a slight modification the network of Fig. 2(a) indicates the models are not equivalent. Consider an additional vertex in this dataset, as shown in Fig. 2(b). The addition barely changes the network's topology, but the two systems behave differently for the transitory domination in the central vertex, as shown in Fig. 3(b). In the edge-centric system, the proportion of particles in the central vertex alternates around half positive and half negative. In the vertex-centric system, the central vertex is dominated by positive-labeled particles. This difference in such simple example motivates us to investigate the advantages of the edge-centric model over the vertex-centric model.

8

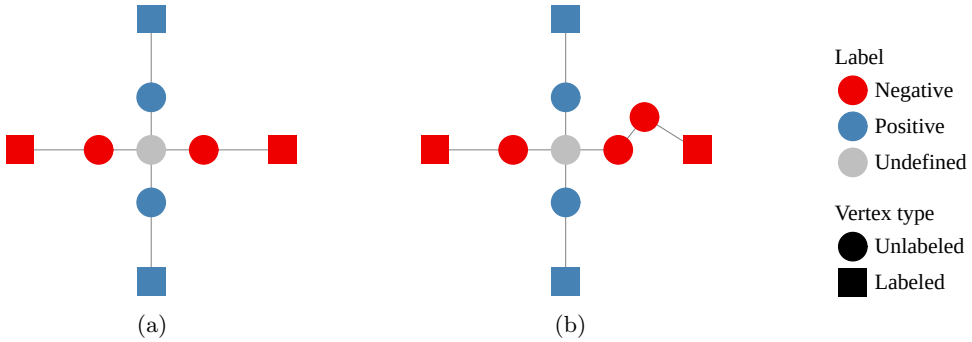


Fig. 2 Two networks, with (a) 9 vertices and (b) 10 vertices, are the simplest case that shows the differences between the results from the competition processes of Edge Domination System (EDS) and Vertex Domination System (VDS). Vertices with the same label shape either a vertical or a horizontal pattern, and each vertex connects to its nearest neighbor. The input is the network with four vertices labeled while the remaining is unlabeled. The central vertex has no label for reference.

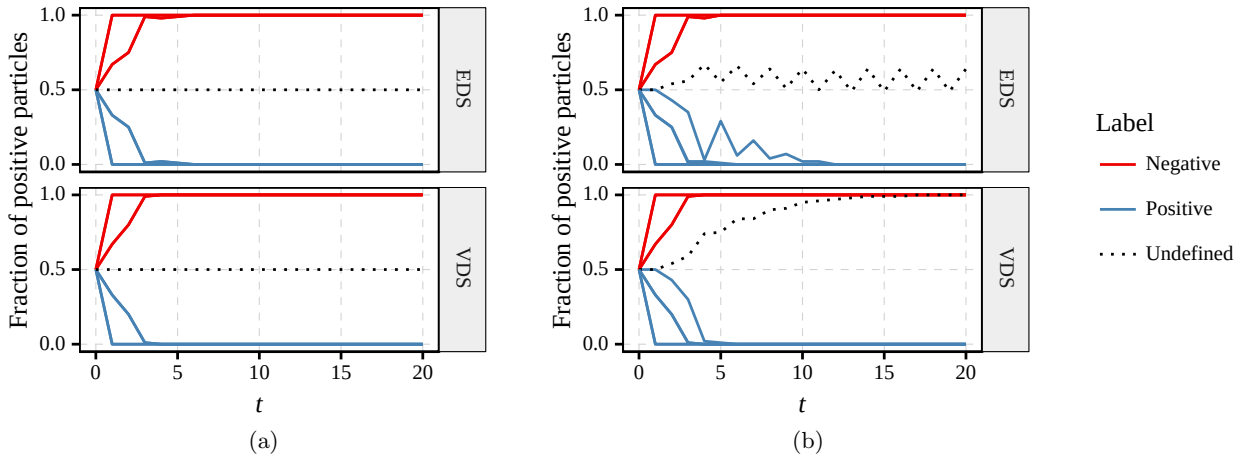


Fig. 3 Comparison of the dynamics of Edge Domination System (EDS) and Vertex Domination System (VDS). Each line is the fraction of *positive* particles in one vertex over time, for input networks with (a) 9 vertices and (b) 10 vertices. The fraction of positive particles in central vertex, which has no label defined, differs between the two systems.

3.2.2 Classification error

We compare the classification error of the systems for increasing levels of difficulty. The input networks are realizations of the model introduced in Section 2.2. The error is obtained from 30 independently sampled networks $G(250,k,q,b)$ for each $k \in \{6,8,\ldots,12\}$, $q \in \{0.8,0.9,\ldots,1.4\}$, and $b \in \{1,2,3\}$. We fixed $\lambda = 1$ and the fraction of initially labeled vertices at 10%.

We start both systems with the population size

$$n_i^c(0) = \frac{\deg v_i}{2|\mathscr{E}|} \tag{21}$$

independently of label c. To classify an unlabeled vertex v_i, we adopt the original strategy in Verri, Urio, and Zhao [4] for the EDS at time $\tau = 100$, $y_i = \arg\max_c |\mathscr{E}(\mathscr{N}_i^c(\tau))|$, where $|\mathscr{E}(\mathscr{N}_i^c(\tau))|$ stands for the number of edges in the neighborhood of v_i in $G^c(\tau)$. In VDS, we just consider the dominating label of the vertex at time $\tau = 100$, that is, $y_i = \arg\max_c n_i^c(\tau)$.

9

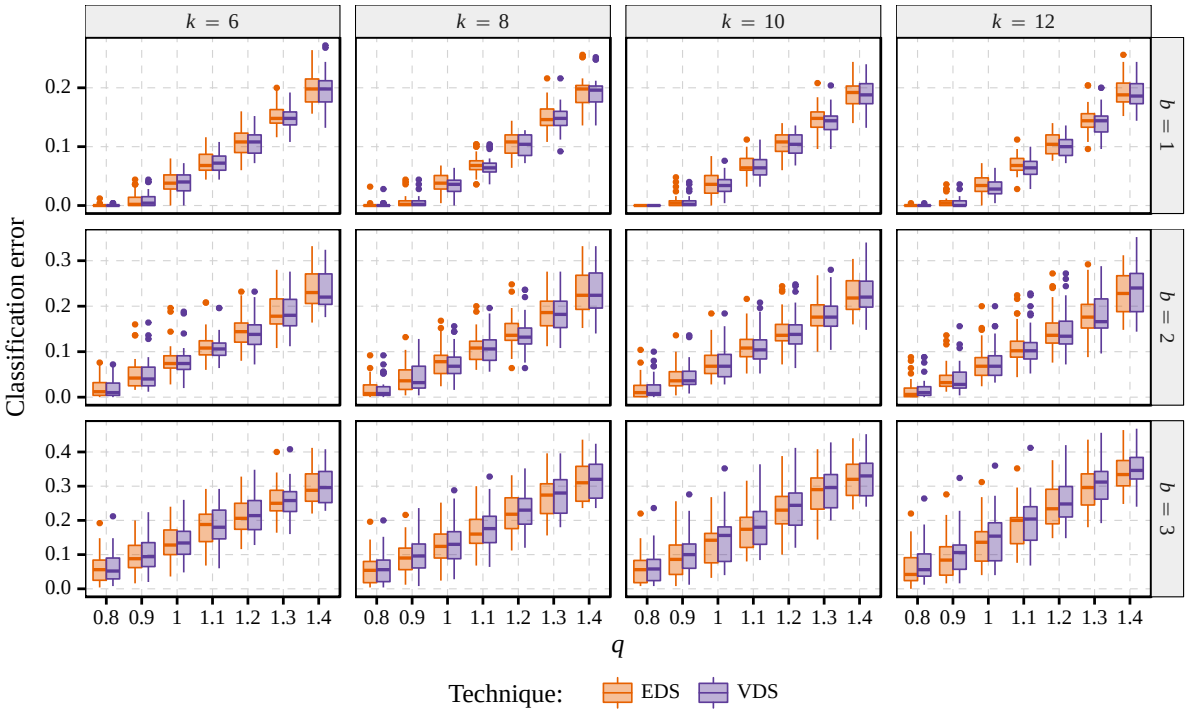


Fig. 4 Classification error of Edge Domination System (EDS) and Vertex Domination System (VDS) for different $G(250,k,q,b)$ networks with 10% of vertices initially labeled.

As shown in Fig. 4, although systems have similar results, EDS slightly outperforms VDS in some cases, especially when the overlapping area is large, $b=3$ and $q>1$. As q and b increase the problem becomes harder, and the systems obtain higher error rates. VDS performs better without overlap, $q<1$, but deteriorates faster while EDS benefit most from the increasing number of edges.

The lower error rate of EDS in problems with large overlap supports the hypothesis that EDS system obtains more information of the border than VDS. Next, we address this hypothesis.

3.2.3 Overlap prediction error

In [4], EDS shows the ability to recognize data items of different labels sharing the same attribute space. We question whether VDS has this capacity as well. Similar to out approach in the toy example in Section 3.2.1, we use the fraction of positive-labeled particles in each vertex to predict the fraction of positive-labeled neighbors of each vertex.

Let $\mathscr{P}_i$ be the set of positive neighbors of the vertex v_i, we use the root-mean-square error

$$\mathrm{RMSE}\left(\left[\frac{n_i^+(\tau=100)}{n_i^+(\tau=100)+n_i^-(\tau=100)}\right]_i, \left[\frac{|\mathscr{P}_i|}{\deg v_i}\right]_i\right),\ i\in\mathscr{V}\setminus\mathscr{L} \tag{22}$$

to assess the effectiveness of the overlap prediction. The error is obtained from the same networks $G(250,k,q,b)$ and vertices initially labeled, similar to the experiment in Section 3.2.2.

Unlike the results of the classification error, EDS consistently outperforms VDS in all cases. Figure 5 summarizes the results, showing EDS retains more information than VDS. But, the classification mechanism proposed in [4] fails at taking advantage of the extra information.

Moreover, a vertex tends to be totally dominated by either positive or negative particles in VDS.

10

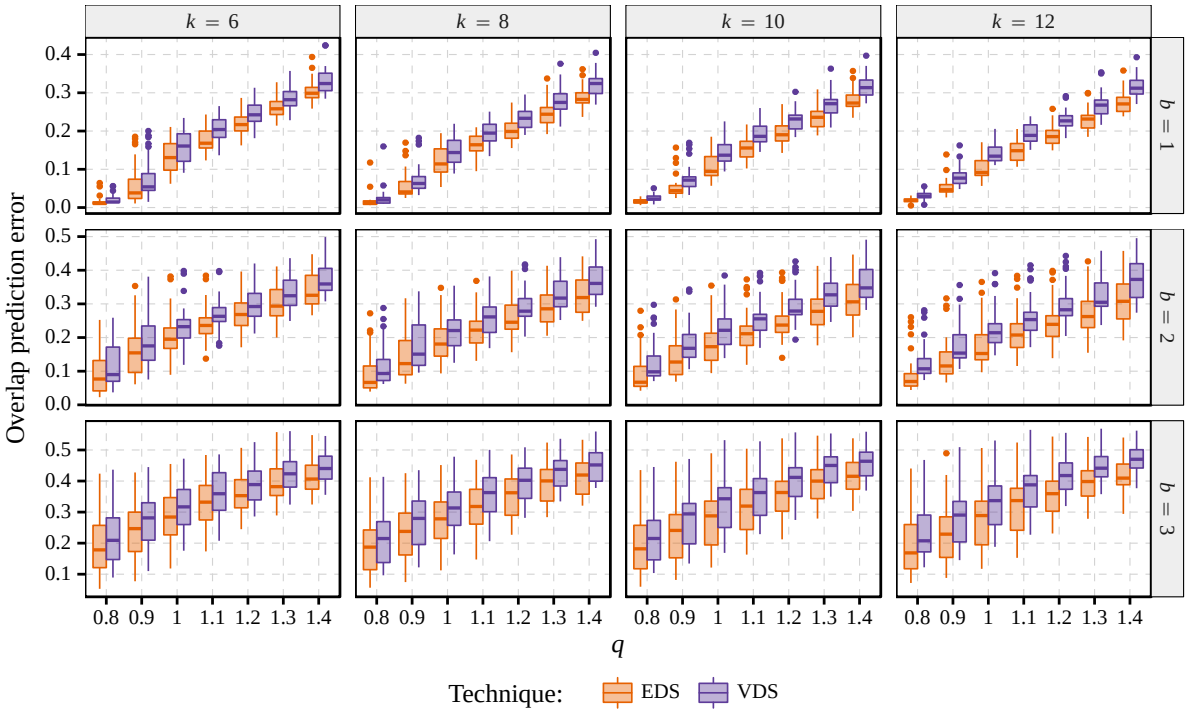


Fig. 5 Comparison of the root-mean-square error of the overlap prediction obtained from Edge Domination System (EDS) and Vertex Domination System (VDS) in different networks $G(250,k,q,b)$, with 10% of vertices initially labeled.

To study this property, we define the entropy $h_i(t)$ of vertex v_i at time t as

$$h_i(t) = -\sum_{c\in\{+,-\}} \frac{n_i^c(t)}{n_i^+(t)+n_i^-(t)} \log_2\left(\frac{n_i^c(t)}{n_i^+(t)+n_i^-(t)}\right). \tag{23}$$

At one extreme, entropy $h_i(t)=0$ means the vertex v_i has the same population of positive and negative particles at time t. At another extreme, entropy $h_i(t)=1$ indicates that only one kind of particles is present in vertex v_i at time t.

The entropy of the vertices in the VDS are extremely low, as shown in Fig. 6. Conversely, the larger is the overlapping area, the higher the entropy is in EDS. We believe this increasing behavior corresponds to the detection of larger borders. A high entropy also suggests that particles propagate farther, reaching larger regions dominated by rival particles. As a result, a greater exploration behavior emerges, and particles are more likely to benefit from the network's topology.

4 Convergence on classification tasks

In the previous experiments, we assessed the classification accuracy of each system in learning problems, which is an important feature to determine the effectiveness in semi-supervised learning tasks. Besides a good accuracy, another important aspect of effectiveness is the time required for an algorithm to solve, because the time complexity order restricts the applicability of a learning algorithm.

Two aspects determine the time necessary for both systems solving a problem. First, both VDS and EDS have linear time complexity regarding the number of vertices in the input network. Second, the number of iterations required for the systems to reach the convergent state. In such case, the complexity

11

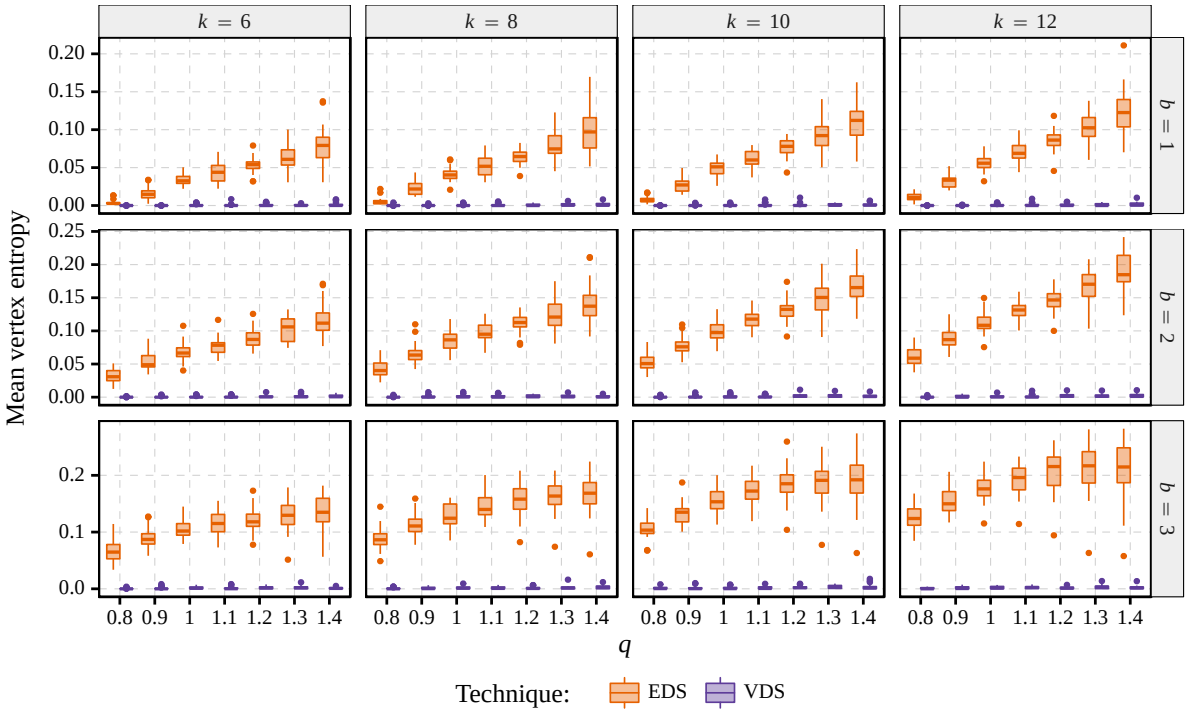


Fig. 6 Mean vertex entropy in Edge Domination System (EDS) and Vertex Domination System (VDS) at time $\tau = 100$ for different networks $G(250,k,q,b)$ (the same networks sampled for the previous experiment) with 10% of vertices initially labeled.

order might be much higher if the convergence time is not sublinear regarding the input. We address the convergence of classification result in both systems at the maximum competition setting ($\lambda = 1$).

4.1 Modeling with maximum competition

Both systems are further simplified fixing the competition parameter in $\lambda = 1$. Considering two labels, $+$ and $-$, we rewrite both systems to contain a variable explicitly related to the classification output. We restrict the initial conditions of both systems, eliminating the conditional terms in Eq. (3) and Eq. (4). Two requirements over the network are also necessary: every vertex should be connected with at least one unlabeled vertex, and two labeled vertices may be connected only if they share the same label. Mathematically,

$$\begin{cases} \sum\limits_{j\,|\,v_j \in \mathscr{V}\setminus\mathscr{L}} a_{ij} > 0, & \text{for all } i \in \{1,\ldots,|\mathscr{V}|\}, \text{ and} \\ a_{ij} = 0 & \text{if } v_i, v_j \in \mathscr{L} \text{ and } y_i \neq y_j. \end{cases} \tag{24}$$

4.1.1 Edge Domination System

In EDS, the classification result depends on the unfoldings, which are convergent if

$$\lim_{t\to\infty} \arg\max_{q} \left(n_{ij}^{q}(t) + n_{ji}^{q}(t) \right) = c \tag{25}$$

for $c = +$ or $c = -$. To further simplify EDS, we replace the nonzero entries of the transitory domination matrix $N^c(t)$ by a set of variables $s_{ij}(t)$ that represent the probability of a positive-labeled particle

surviving at time t given it decided to move from vertex v_i to vertex v_j. Let

$$s_{ij}(t) = \frac{n_{ij}^{+}(t) + n_{ji}^{+}(t)}{n_{ij}^{+}(t) + n_{ji}^{+}(t) + n_{ij}^{-}(t) + n_{ji}^{-}(t)} \tag{26}$$

and imposing the restrictions

$$\begin{cases} n_i^c(0) = 0 & \text{if } v_i \in \mathscr{L} \text{ and } y_i \neq c, \\ n_i^c(0) > 0 & \text{otherwise,} \end{cases} \tag{27}$$

$$\sum_i n_i^c(0) = 1, \text{ and}$$

$$n_{ij}^c(0) = \begin{cases} 0 & \text{if } v_i \in \mathscr{L} \text{ and } y_i \neq c, \\ 0 & \text{if } v_j \in \mathscr{L} \text{ and } y_j \neq c, \\ 1 & \text{otherwise,} \end{cases} \tag{28}$$

we rewrite the evolution function of EDS as

$$\phi : \begin{cases} n_i^{+}(t+1) = \rho_i^{+}\left(1 - \sum_j n_j^{+}(t)\right) + \sum_j n_j^{+}(t)\frac{a_{ji}}{\deg v_j} s_{ji}(t) \\ n_i^{-}(t+1) = \rho_i^{-}\left(1 - \sum_j n_j^{-}(t)\right) + \sum_j n_j^{-}(t)\frac{a_{ji}}{\deg v_j}\left[1 - s_{ji}(t)\right] \\ s_{ij}(t+1) = \dfrac{\left(n_i^{+}(t)\frac{a_{ij}}{\deg v_i} + n_j^{+}(t)\frac{a_{ji}}{\deg v_j}\right) s_{ij}(t)}{\left(n_i^{+}(t)\frac{a_{ij}}{\deg v_i} + n_j^{+}(t)\frac{a_{ji}}{\deg v_j}\right) s_{ij}(t) + \left(n_i^{-}(t)\frac{a_{ij}}{\deg v_i} + n_j^{-}(t)\frac{a_{ji}}{\deg v_j}\right)\left[1 - s_{ij}(t)\right]} \end{cases}, \tag{29}$$

with the initial condition

$$s_{ij}(0) = \begin{cases} 0 & \text{if } v_i \in \mathscr{L} \text{ and } y_i = -, \\ 0 & \text{if } v_j \in \mathscr{L} \text{ and } y_j = -, \\ 1 & \text{if } v_i \in \mathscr{L} \text{ and } y_i = +, \\ 1 & \text{if } v_j \in \mathscr{L} \text{ and } y_j = +, \\ 0.5 & \text{otherwise.} \end{cases} \tag{30}$$

The unfoldings $G^c(t) = (\mathscr{V}, \mathscr{E}^c(t))$ are also rewritten in terms of s_{ij}, that is,

$$\mathscr{E}^{+}(t) = \left\{(i,j) \middle| s_{ij}(t) + s_{ji}(t) > 1\right\}, \text{ and } \mathscr{E}^{-}(t) = \left\{(i,j) \middle| s_{ij}(t) + s_{ji}(t) < 1\right\}. \tag{31}$$

To find out the equations for $s_{ij}(t+1)$, $n_{ij}^{+}(t+1)$, and $n_{ij}^{-}(t+1)$, we first simplify Eq. (3) under the previous assumptions.

Lemma 2. *Assuming $\lambda = 1$, the network requirements in Eq.* (24) *and the initial conditions in Eqs.* (27) *and* (28)*,*

$$p_{ij}^c(X(t)) = \frac{a_{ij}}{\deg v_i} \frac{n_{ij}^c(t) + n_{ji}^c(t)}{n_{ij}^{+}(t) + n_{ji}^{+}(t) + n_{ij}^{-}(t) + n_{ji}^{-}(t)},$$

for all time t and labels $c \in \{+,-\}$ in EDS*.*

Proof. Let $\mathsf{sink}_c\, v_i$ be a logical function that yields true if v_i is a sink for particles of team c. We observe the evolution of $n_i^c(t)$, $n_{ji}^c(t)$, and $n_{ij}^c(t)$, in vertices v_i such that $\mathsf{sink}_c\, v_i$ holds true. From Eqs. (27)

13

and (28), $n_i^c(0) = n_{ij}^c(0) = 0$, for all j. For any $t \geq 0$,

$$n_i^c(t+1) = \sum_j n_j^c(t) p_{ji}^c(t) + \rho_i^c \left(1 - \sum_j n_j^c(t)\right) = 0,$$

$$n_{ji}^c(t+1) = n_j^c(t) p_{ji}^c(t) = 0,$$

since $p_{ji}^c(t) = \rho_i^c = 0$ if v_i is a sink for particle of team c, see Eq. (3). Once $n_i^c(t) = 0$ for any time t,

$$n_{ij}^c(t+1) = n_i^c(t) p_{ij}^c(t) = 0.$$

For the other vertices v_i and v_j, such that $\mathsf{sink}_c\, v_i$ and $\mathsf{sink}_c\, v_j$ holds false, we show that $n_i^c(t), n_{ji}^c(t) > 0$ by mathematical induction. Equations (27) and (28) guarantee these inequalities for $t = 0$. Assuming they hold true for t, we extend to $t+1$:

$$n_i^c(t+1) = \sum_j n_j^c(t) p_{ji}^c(t) + \rho_i^c \left(1 - \sum_j n_j^c(t)\right) \geq \sum_j n_j^c(t) p_{ji}^c(t) = \sum_j n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \sigma_{ji}^c(X(t))\right) = \sum_{j \,|\, \neg \mathsf{sink}_c\, v_j} n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \sigma_{ji}^c(X(t))\right) > 0,$$

$$n_{ji}^c(t+1) = n_j^c(t) p_{ji}^c(t) = n_j^c(t) \frac{a_{ji}}{\deg v_j} \left(1 - \sigma_{ji}^c(X(t))\right) > 0,$$

since

$$\sigma_{ji}^c(X(t)) = 1 - \frac{n_{ij}^c(t) + n_{ji}^c(t)}{n_{ij}^+(t) + n_{ji}^+(t) + n_{ij}^-(t) + n_{ji}^-(t)} < 1,$$

for all i, j such that $\neg\, \mathsf{sink}_c\, v_i$ and $\neg\, \mathsf{sink}_c\, v_j$, and every vertex has a non-labeled neighbor.

From these results, $n_{ij}^+(t) + n_{ji}^+(t) + n_{ij}^-(t) + n_{ji}^-(t) > 0$, as a vertex cannot be a source for both positive and negative particles. Thus, we drop the conditional in Eq. (4),

$$p_{ij}^c(X(t)) = \begin{cases} 0 & \text{if } \mathsf{sink}_c\, v_j, \\ \dfrac{a_{ij}}{\deg v_i} \dfrac{n_{ij}^c(t) + n_{ji}^c(t)}{n_{ij}^+(t) + n_{ji}^+(t) + n_{ij}^-(t) + n_{ji}^-(t)} & \text{otherwise.} \end{cases}$$

However, $n_{ij}^c(t) + n_{ji}^c(t) = 0$ if $\mathsf{sink}_c\, v_j$, eliminating the need of the conditional.

From Lemma 2, it is straightforward derive the simplified system. Moreover, the restriction of two rival sinks not being connected guarantees the coherence of the initial state.

4.1.2 Vertex Domination System

Analogously to the previous simplification, we introduce to VDS a set of variables $s_i(t)$ that is the probability of a positive-labeled particle surviving at time t given it decided to move to vertex v_i independently of its origin. Let

$$s_i(t) = \frac{n_i^+(t)}{n_i^+(t) + n_i^-(t)} \tag{32}$$

and considering the restriction in Eq. (27), the VDS is rewritten as

$$\begin{cases} n_i^+(t+1) = \rho_i^+ \left(1 - \sum_j n_j^+(t)\right) + s_i(t) \sum_j n_j^+(t) \frac{a_{ji}}{\deg v_j} \\ n_i^-(t+1) = \rho_i^- \left(1 - \sum_j n_j^-(t)\right) + [1 - s_i(t)] \sum_j n_j^-(t) \frac{a_{ji}}{\deg v_j} \\ s_i(t+1) = \frac{s_i(t) \sum_j n_j^+(t) \frac{a_{ji}}{\deg v_j}}{s_i(t) \sum_j n_j^+(t) \frac{a_{ji}}{\deg v_j} + [1 - s_i(t)] \sum_j n_j^-(t) \frac{a_{ji}}{\deg v_j}} \end{cases}, \tag{33}$$

with the initial condition

$$s_i(0) = \begin{cases} 0 & \text{if } v_i \in \mathscr{L} \text{ and } y_i = -, \\ 1 & \text{if } v_i \in \mathscr{L} \text{ and } y_i = +, \\ 0.5 & \text{otherwise.} \end{cases} \tag{34}$$

The classification mechanism adopted in this paper for VDS becomes

$$y_i = \begin{cases} + & \text{if } s_i(t) > 0.5, \\ - & \text{otherwise.} \end{cases} \tag{35}$$

The proof is analogous to the simplification of EDS.

4.2 Analytical study

The introduced variables s_{ij} and s_i are directly related to the classification results of the EDS and VDS, respectively. Their convergence implies the convergence of the classification results.

In EDS, we show that once an edge is completely dominated by particles of the same team, its domination level will not change anymore. Let $s_{ij}(t+1) = s_{ij}(t) = s_{ij}$ at some time t and some i, j. Thus,

$$s_{ij} = \frac{\left(n_i^+(t) \frac{a_{ij}}{\deg v_i} + n_j^+(t) \frac{a_{ji}}{\deg v_j}\right) s_{ij}}{\left(n_i^+(t) \frac{a_{ij}}{\deg v_i} + n_j^+(t) \frac{a_{ji}}{\deg v_j}\right) s_{ij} + \left(n_i^-(t) \frac{a_{ij}}{\deg v_i} + n_j^-(t) \frac{a_{ji}}{\deg v_j}\right) [1 - s_{ij}]} \tag{36}$$

implies that $s_{ij} = 0$ and $s_{ij} = 1$ are fixed points.

The same holds for the vertex dominance in VDS. Let $s_i(t+1) = s_i(t) = s_i$ for some time t and for all i. Thus,

$$s_i = \frac{s_i \sum_j n_j^+(t) \frac{a_{ji}}{\deg v_j}}{s_i \sum_j n_j^+(t) \frac{a_{ji}}{\deg v_j} + [1 - s_i] \sum_j n_j^-(t) \frac{a_{ji}}{\deg v_j}} \tag{37}$$

implies that $s_i = 0$ and $s_i = 1$ are fixed points.

4.3 Experimental simulations

Regarding the convergence speed, we experimentally compare the time the variables s_{ij} and s_i take to converge. The convergence time is obtained from 30 independently sampled networks $G(n,k,q,b)$ for every $n \in \{50, 100, \ldots, 500\}$, $k \in \{6, 7, \ldots, 10\}$, $q \in \{0.8, 1.2\}$, and $b \in \{1, 2, 3\}$. We set the fraction of initially labeled vertices at 10%.

15

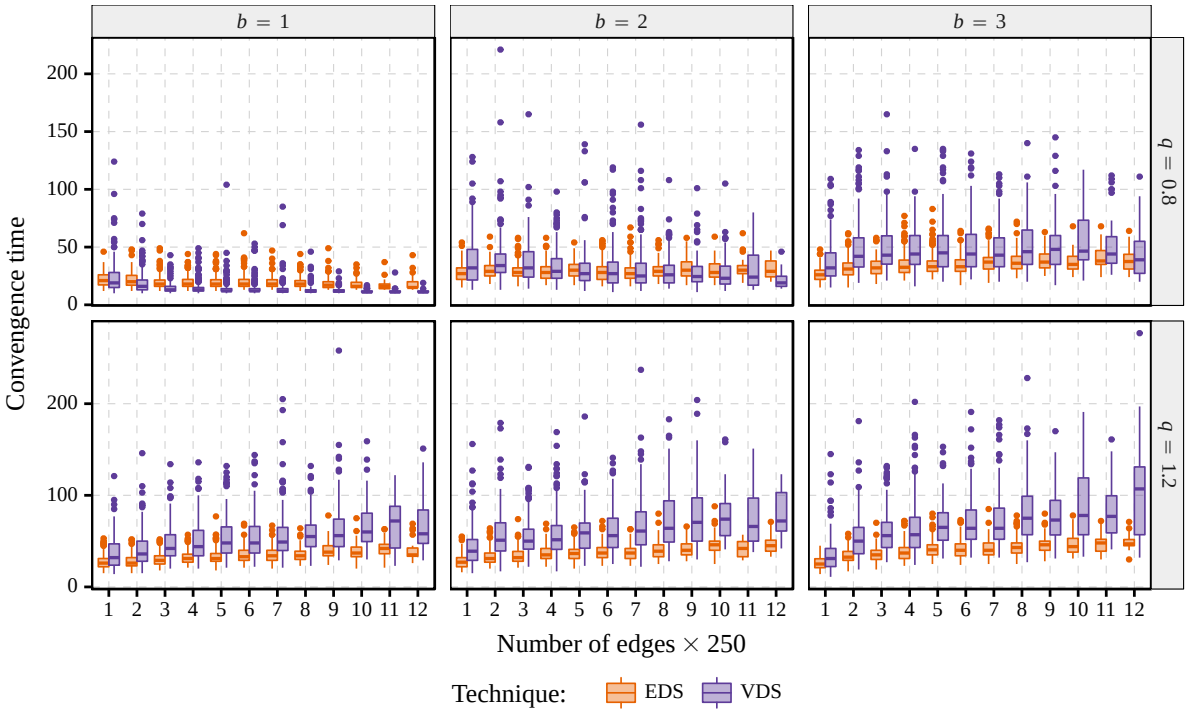


Fig. 7 Convergence time of variables s_{ij} in Edge Domination System (EDS) and s_i in Vertex Domination System (VDS) for increasing number of edges in networks $G(n,k,q,b)$ with 10% of vertices initially labeled.

We start both systems with population size

$$n_i^c(0) = \begin{cases} 0 & \text{if } v_i \in \mathscr{L} \text{ and } y_i \neq c, \\ \left|\mathscr{V} \setminus \{v_j | v_j \in \mathscr{L} \text{ and } y_j \neq c\}\right|^{-1} & \text{otherwise,} \end{cases} \tag{38}$$

to comply with the simplification requirements.

Figure 7 depicts the convergence time by the number of edges in the network. The convergence time differs significantly according to the overlap of the networks.

For $q = 0.8$, few vertices neighbor rival vertices, resulting in less competition between rival teams. VDS benefits from the increasing number of edges and the variables s_i take fewer iterations to converge than the variables s_{ij} of EDS. We speculate that all particles generated in a source reach all the vertices of the same team faster because the networks' diameters are decreased by the larger number of edges.

With larger overlapping areas, $q = 1.2$, more competition occurs due to a larger number of edges, causing both systems to need more time before they converge. In this case, the classification results from EDS converge faster than those of VDS, which reinforces the discussion in Section 3.2.2.

EDS also offers a much lower deviance for the time of convergence. The variance for VDS increases significantly compared to EDS in some cases. Nevertheless, the convergence time in both systems is sublinear for the number of edges in the network. Machine learning models can employ the convergent results of these systems with subquadratic time complexity order.

5 Convergence time and classification error in benchmark datasets

We explored the existence of a trade-off between convergence time and classification error when comparing both systems in well-known real and artificial datasets. The Chapelle's benchmark [7] encompasses

16

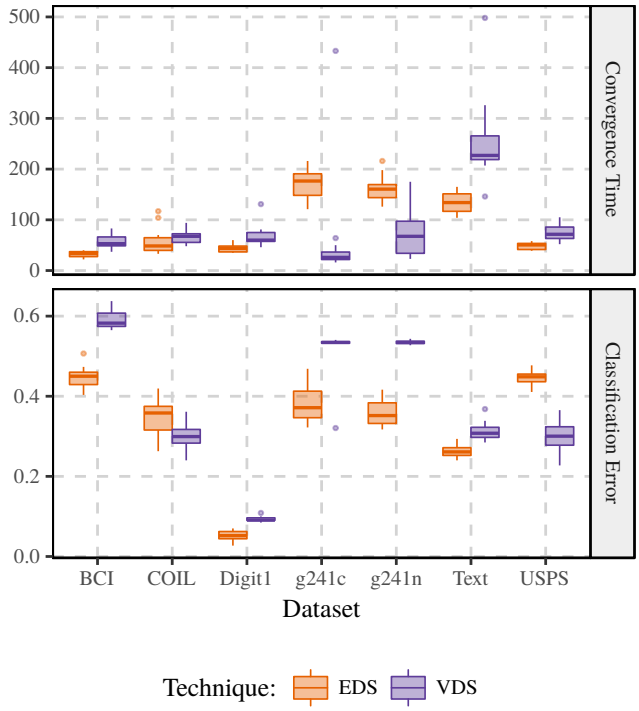


Fig. 8 Convergence time and classification accuracy error of vertex-based and edge-based domination system in datasets of the Chapelle's benchmark.

12 datasets, each one corresponding to a different situation or domain knowledge. We arbitrarily selected 7 datasets with 1500 points and 2 classes. (Two exceptions, BCI has 400 points, and COIL has 6 classes.) For each dataset, we use 12 distinct sets of 100 labeled points. The networks are obtained from the k-Nearest Neighbors graph method, whose hyperparameter k was picked following the best-found in [4] for the same network construction method.

Figure 8 shows the results of this experiment considering the systems with maximum competition. EDS takes longer to converge than VDS in two datasets, g241c and g241n, which is compensated by smaller classification errors. Conversely, in USPS, an imbalanced dataset, and COIL, a six-class dataset, EDS converges faster while obtaining a larger classification error. In the remanining datasets—BCI, Digit1, and Text—EDS has faster convergence and lower classification error.

Since there is no conclusive evidence of a trade-off between convergence time and accuracy, we hypothesize the two systems are indeed different, and they depend on different characteristics of the network's topology, which depends on the nature of the datasets. Such dependency shall be studied in future investigations.

6 Conclusion

We studied the advantages of an edge-centric collective dynamics in semi-supervised learning tasks. The previously proposed Edge Domination System (EDS) [4] is a particle competition system that can be used to solve data classification tasks. We present a similar but vertex-based version of the system, Vertex Domination System (VDS), and studied their differences.

We showed that the original system could be simplified without impairing its performance in learning problems. The models offer similar performance in classification and the same time complexity order.

However, EDS not only acquires more information about the overlapping area with mixed rival vertices, but also has better exploration behavior; see Section 3.2.3. It has advantages regarding the running time as well. The average convergence time of the classification results in EDS is lower than in VDS. Moreover, while the standard deviation of running time is lower in EDS, in VDS it increases with the number of edges; see Section 4.3.

EDS is already shown to achieve good classification accuracy in real applications [4]. With maximum competition, it usually converges faster than VDS while maintaining the same accuracy—if not better– in most of the cases; see Section 5. Thus, the evidence points towards favoring EDS over VDS.

In future works, we will propose a new classification mechanism for EDS that takes more advantage over the produced unfoldings. We speculate that the newer classification mechanism will increase the classification accuracy of EDS. We also intend to estimate the expected time of convergence analytically.

Acknowledgments

This research was supported by the São Paulo State Research Foundation (FAPESP), the Coordination for the Improvement of Higher Education Personnel (CAPES), and the Brazilian National Research Council (CNPq).

References

[1] Gueleri, R.A., Cupertino, T.H., de Carvalho, A.C.P.L.F., and Zhao, L. (2014), A flocking-like technique to perform semi-supervised learning, *International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1579–1586, DOI: 10.1109/IJCNN.2014.6889434.

[2] Zhang, Z., Long, K., Wang, J., and Dressler, F. (2014), On swarm intelligence inspired self-organized networking: Its bionic mechanisms, designing principles and optimization approaches, *IEEE Commun. Surveys Tuts.*, **16**(1), pp. 513–537.

[3] Breve, F., Quiles, M.G., and Zhao, L. (2015), Interactive image segmentation using particle competition and cooperation, *International Joint Conference in Neural Networks, Proceedings*, IEEE, pp. 1–8.

[4] Verri, F.A.N., Urio, P.R., and Zhao, L. (2016), Network Unfolding Map by Vertex-Edge Dynamics Modeling, *IEEE Transactions on Neural Networks and Learning Systems*, **PP**(99), pp. 1–14, ISSN: 2162-237X, DOI: 10.1109/TNNLS.2016.2626341.

[5] Silva, T.C. and Zhao, L. (2016), Machine Learning in Complex Networks, 1st ed., Springer, p. 331, ISBN: 978-3-319-17290-3, DOI: 10.1007/978-3-319-17290-3.

[6] Urio, P.R., Verri, F.A.N., and Zhao, L. (2016), Features of Edge-Centric Collective Dynamics in Machine Learning Tasks, *International Conference on Nonlinear Science and Complexity, Proceedings*.

[7] Chapelle, O., Schölkopf, B., and Zien, A., eds. (2006), Semi-Supervised Learning, MIT Press: Cambridge, MA, p. 524, ISBN: 9780262033589, DOI: 10.7551/mitpress/9780262033589.001.0001.

CHAPTER

4

RANDOM WALK IN FEATURE–SAMPLE NETWORKS FOR SEMI-SUPERVISED CLASSIFICATION

© 2016 IEEE. Reprinted, with permission, from Filipe A. N. Verri, Liang Zhao, “Random Walk in Feature–Sample Networks for Semi-Supervised Classification”, Proceeding of the 5th Brazilian Conference on Intelligent Systems (BRACIS), Recife, Brazil, 2016.

Contribution statement

F.A.N. Verri conceived and developed the research. F.A.N. Verri and L. Zhao wrote the manuscript. L. Zhao supervised the project.

Random Walk in Feature–Sample Networks for Semi-Supervised Classification

Filipe Alves Neto Verri
Institute of Mathematical and Computer Sciences
University of São Paulo – São Carlos, SP, Brazil
School of Electrical, Computer and Energy Engineering
Arizona State University – Tempe, AZ, USA
E-mail: filipeneto@usp.br

Liang Zhao
Ribeirão Preto School of Philosophy, Science and Literature
University of São Paulo
Ribeirão Preto, SP, Brazil
E-mail: zhao@usp.br

***Abstract*—Positive-unlabeled learning is a semi-supervised task in which only some positive-labeled and many unlabeled samples are available. The goal of its transductive setting is to label all unlabeled data at once. In this paper, we developed a technique to grade positive-class pertinence levels of each sample, and we interpret the grades to classify the unlabeled ones. In our method, a sparse binary matrix represents the input data, which determines the feature–sample network whose vertices represent samples and attributes. The limiting probabilities of a random walk in the network estimate the pertinence levels. The results are evaluated regarding both class discrimination and classification accuracy. Computer simulations reveal that our model performs well in positive-unlabeled learning, especially with few labeled samples. Notably, the outcomes compare to the results from supervised methods, which profit from most data labeled. Additionally, the technique has linear time and space complexity if the input dataset is already in a sparse representation. The low computational cost of the construction and update of the feature–sample network allows for extensions of the technique to several learning problems, including online learning and dimensionality reduction.**

***Index Terms*—Semi-supervised classification, complex networks, positive-unlabeled learning, random walk.**

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. Published version: `http://ieeexplore.ieee.org/document/7839592/`. DOI 10.1109/BRACIS.2016.051.

I. Introduction

Several machine learning techniques can learn not only from labeled data but also from unlabeled instances. Such models belong to the semi-supervised learning paradigm [1]. Partially labeled datasets are plentiful because more information is generated in our world than we can label by hand.

Many semi-supervised techniques model the input data either as graphs or as complex networks [2]–[5]. In both cases, each vertex usually represents a data sample, and an edge exists if its endpoints satisfy a predefined affinity rule. In graph-based techniques, an optimization process propagates the labels from labeled vertices to the unlabeled ones. In techniques based on complex networks, the algorithm classifies data by exploring topological and evolution properties of a certain collective dynamics. These methods provide a flexible and robust learning process.

A special scenario of semi-supervised classification comprises only few positive-labeled and many unlabeled samples; this problem is called positive-unlabeled (PU) learning [6]–[8]. The goal of PU learning tasks is either to build a classifier that discriminates positive and negative samples or to label all unlabeled input samples at once. Techniques that accomplish the former are inductive; while the ones that achieve the latter are transductive. Some graph-based techniques have been proposed [9], [10], but no complex-network approach exists.

We introduce a transductive PU learning technique based on complex networks with steps: *a)* The input dataset is converted into a sparse binary representation. (We use the terms feature and attribute interchangeably.) *b)* The binary representation models a network in which a vertex represents either a sample or an attribute. *c)* A random walk process is performed over the network taking into account the labeled samples; this process is a discrete Markov chain with states associated with the network's vertices. *d)* The positive-class pertinence level of each unlabeled sample is calculated from the limiting distribution of the Markov chain. *e)* Unlabeled samples are classified using the positive-class pertinence and with knowledge of the positive-class prior probability.

The model is evaluated regarding both class discrimination and classification accuracy. To assess how well it discriminates positive from negative samples, we introduce a measurement of class discrimination and a fair baseline algorithm to compare with. Concerning the classification accuracy, the results are compared with state-of-the-art techniques.

The proposed scheme excels on PU learning problems. Compared with PU classifiers, the technique surpasses other methods if the prior probabilities are known. Compared with supervised classifiers, the results are similar, even though those classifiers profit from much more labeled samples available during the training step. The research shows potential for a broad range of applications.

The rest of this paper is organized as follows. Sections II and III describe the proposed model and its computational complexity. In Section IV, computer simulations illustrate the learning process and assess its performance. Finally, Sections V and VI discuss and conclude this paper.

II. Model Description

Let $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ be the dataset where each data item $\vec{x}_i$ is either a *positive* or a *negative* sample. Examples $\vec{x}_i$ are positive labeled for all $i \in \mathcal{P}$, and the remaining samples are unlabeled. The prior probability P^+ of the positive class is known. Our goal is to estimate a positive-class pertinence level $f(\vec{x}_i)$ of each unlabeled data $\vec{x}_i$, $i \notin \mathcal{P}$, and to classify them using the priors.

In the next subsections, we explain the steps of our learning algorithm to solve the stated problem.

A. Conversion to the Binary Sparse Representation

In our model, the input dataset must satisfy three requirements: each sample is a binary feature vector, an attribute with value 1 indicates the *presence* of a characteristic of that data instance, and the similarity of two samples depends on the number of shared characteristics but not on mismatching and absent features. Such requirements likely cause feature vectors to be sparse.

Most of the datasets are easily converted to this representation. We converted datasets composed of numerical and categorical attributes as follows: *a)* Replace each categorical feature $x \in \{X_1, \ldots, X_K\}$ of each sample by a binary feature vector $\{b_k\}_{k=1,\ldots,K}$ such that $b_k = 1 \iff x = X_k$. The sparsity of the new features is proportional to the number of possible categorical values K. *b)* Discretize numerical attributes disregarding the class information. The most common approaches are by equal interval width and by equal frequency. *c)* Convert the categorical features obtained from discretization to vectors of binary features as well.

B. Construction of the Feature–Sample Network

We derive a complex network from the binary dataset. Let $\mathcal{B} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ be the set where each element $\vec{x}_i$ is a binary feature vector $(x_{i1}, \ldots, x_{iM}) \in \{0,1\}^M$. The *Feature–Sample Network* $\mathcal{G}$ is the bipartite complex-network whose edges associates samples and features of the dataset $\mathcal{D}$. A simple, unweighted, undirected graph $(\mathcal{V}, \mathcal{E})$ represents such network. The vertex set $\mathcal{V}$ is $\{v_1, \ldots, v_N, v_{N+1}, \ldots, v_{N+M}\}$ and an edge exists between sample v_i and feature v_{N+j} if $x_{ij} = 1$. The adjacency matrix $A = (a_{ij})$ of $\mathcal{G}$ has elements

$$a_{ij} = \begin{cases} x_{i,j-N} & \text{if } 1 \le i \le N \text{ and } N < j \le N+M, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We suppose that $\mathcal{G}$ has a single connected component. If it is false in an experiment, we consider only the largest connected component that contains all (or most) of the labeled samples.

C. Modeling of the Random Walk Process

The next step is to perform a random walk over the network. This random process is a discrete Markov chain with transition matrix $P = (p_{ij})$, such that p_{ij} is the probability of going from v_i to v_j. We model P to guarantee the existence and uniqueness of the limiting distribution of such Markov chain.

Since the $\mathcal{G}$ is connect, the process is a time-homogeneous and irreducible Markov chain. An irreducible Markov chain has a unique stationary distribution if and only if all states are positive recurrent [11]. Since $\mathcal{G}$ is undirected and finite, all states of a random walk over $\mathcal{G}$ are positive recurrent. Thus, a unique stochastic vector $\vec{\pi}$ exists such that

$$\vec{\pi} P = \vec{\pi}, \quad (2)$$

if $p_{ij} > 0$ for all $a_{ij} = 1$.

We also want to reach the stationary distribution $\vec{\pi}$ from any initial distribution. The limiting distribution of a random walk is reached independently of the initial conditions if the irreducible Markov chain is ergodic, that is, both aperiodic and positive recurrent [11]. Since every state of a bipartite graph has an even period, the limiting distribution of a random walker in $\mathcal{G}$ may never reach the stationary distribution. To achieve an ergodic Markov chain, we include non-zero entries on the main diagonal of P.

Finally, the transition probabilities are

$$p_{ij} = \frac{w_{ij}\nu_j}{\lambda \nu_i}, \quad (3)$$

where $W = (w_{ij})$ such that

$$w_{ij} = \begin{cases} 1 & \text{if } i = j, \\ \beta_i a_{ij} & \text{otherwise,} \end{cases} \quad (4)$$

and $\vec{\nu} = (\nu_1, \ldots, \nu_{N+M})$ is the eigenvector associated with the leading eigenvalue λ of the matrix W. The scaling factor β_i is $\beta > 1$ if $\vec{x}_i$ is a positive-labeled sample. Otherwise, the scaling factor is 1.

D. Estimation of the Positive-class Pertinence Level

The transition matrix P describes a system with the desired behavior: the limiting distribution $\vec{\pi}$ is reached independently of the initial setting; the states associated with features relevant to the positive class will have high stationary probabilities; and the limiting probabilities related to positive samples are expected to be greater than the ones associated with negative samples.

We estimate the positive-class pertinence level $f(\vec{x}_i) = \pi_i$ for all $i \notin \mathcal{P}$. The stochastic vector $\vec{\pi} = (\pi_i)$ is a eigenvector associated with the leading eigenvalue of the matrix P^T. Alternatively, it can be calculated by iterating the system $\vec{\pi}(t+1) = \vec{\pi}(t)P$ with any initial configuration.

E. Classification of the unlabeled samples

We classify a unlabeled sample $\vec{x}_i$, $i \notin \mathcal{P}$, as positive if its pertinence level $f(\vec{x}_i)$ is greater than a certain threshold; which satisfies the expected number positive samples according to the prior probability P^+

The predicted class $c(\vec{x}_i)$ of an unlabeled sample $\vec{x}_i$ is

$$c(\vec{x}_i) = \begin{cases} +1 & \text{if } f(\vec{x}_i) > f^{\text{ordered}}_{\lceil (N - |\mathcal{P}|)P^+ \rceil} \\ -1 & \text{otherwise,} \end{cases} \quad (5)$$

where f^{ordered}_n is the n-th greatest value of $f(\vec{x}_i)$ for all $i \notin \mathcal{P}$.

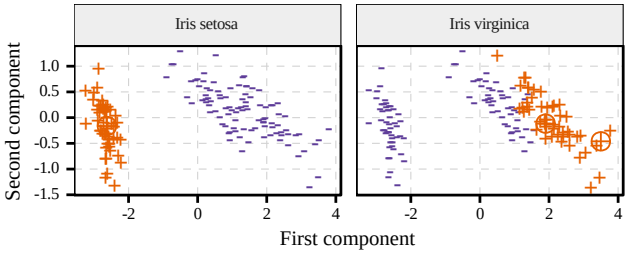


Figure 1. PCA projection of the Iris dataset, with samples of species *Iris setosa* (left-hand plot) and *Iris virginica* (right-hand plot) as the positive class.

III. Time and Space Complexity

We analyze the computational complexity of our algorithm regarding N and D, the number of samples and features.

The conversion of the input dataset into a sparse binary representation takes $\mathcal{O}(DN \log_2 N)$. The worst case scenario happens when all D numerical attributes must be discretized. Since the features along all samples would need to be sorted, the average sorting time adds up to the time complexity.

The construction of the feature–sample network takes $\mathcal{O}(DN)$, since it highly sparse, and the number of edges is *limited* by DN.

Both modeling the process and searching the stationary distribution take $\mathcal{O}(DN)$. The two depend on the choice of the eigenvalue algorithm and the matrix representation. With sparse matrices and iterative algorithms – for example, the power iteration algorithm – the time and space requirement is proportional to DN.

In summary, our method runs in $\mathcal{O}(DN)$ if the input dataset is binary, and in $\mathcal{O}(DN \log_2 N)$ otherwise. Since we only store matrices with up to DN nonzero entries, our method has space complexity $\mathcal{O}(DN)$.

IV. Experimental Simulations

In the following subsections, we provide computer simulations to illustrate the learning process and assess its performance in real-world datasets.

A. Illustrative Example

The UCI Iris dataset [12] comprises 50 samples for each of three species of Iris flower—*Iris setosa*, *Iris virginica*, and *Iris versicolor*. Each data item contains 4 numerical features which stand for measurements of the width and length of the sepals and petals in centimeters.

Similarities between samples of the three classes vary considerably. *Iris setosa* samples are quite distinct from the other two species. *Iris virginica* and *Iris versicolor* samples, on the other hand, share more similarities between themselves. Figure 1 clarify this property, showing the first two principal components of the PCA projection. We consider two input setups: *a) Iris setosa* as the positive class and only a single positive sample labeled; and *b) Iris virginica* as the positive class and two positive samples labeled. The labeled samples are circled in Figure 1. For both scenarios, we describe our learning process step by step.

Table I
Features of labeled samples shown in Figure 1.

Class	Sepal Length	Sepal Width	Petal Length	Petal Width
setosa	5.0	3.4	1.5	0.2
virginica	7.7	2.8	6.7	2.0
virginica	6.2	3.4	5.4	2.3

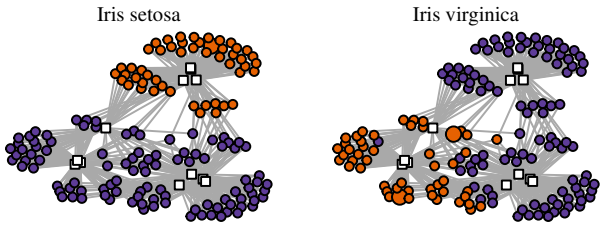


Figure 2. Feature–sample network for Iris dataset with samples of species *Iris setosa* (left-hand side) and *Iris virginica* (right-hand side) as the positive class. Circles are vertices associated with samples and squares with features. Positive samples are highlighted in orange (light).

1) Binary Sparse Representation: We convert the dataset into a sparse binary representation; independently of the labeled samples. Numerical attributes are discretized in 3 intervals by frequency.

The discretized representation reduce the numerical detail while holding sufficient information for the classification task. Tables I and II compare both representations. The first line represent *Iris setosa* samples, and the remaining represent the *Iris virginica* samples. The discretization intervals are below the features names in Table II. Although the discrete representation loses information, the sophistication of the learning process overcomes its simplification.

2) Feature–Sample Network: With the binary representation of data and Equation (1), we construct the feature–sample network. Figure 2 illustrates two networks for the Iris dataset. Circles are vertices of *samples*, and squares are vertices of *features*. The left-hand network, in which *Iris setosa* is the positive class, has positive samples in light orange, and the labeled samples are bigger. The right-hand network, in which *Iris virginica* is the positive class, has the same characteristics.

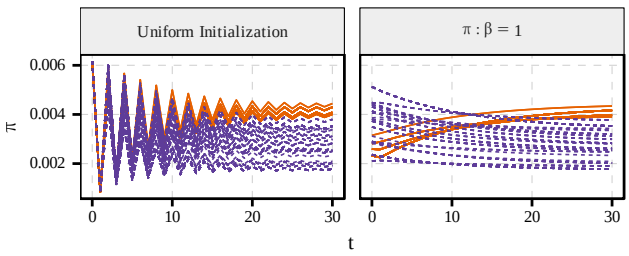


Figure 3. Evolution of the probability distribution with two different initial distribution. The process is modeled from Iris dataset with one labeled sample of class *Iris setosa* and $\beta = 6$. Only values associated with unlabeled samples are shown. Solid orange lines and dashed purple lines are associated with positive and negative samples in that order.

Table II
SPARSE BINARY REPRESENTATION OF THE SAMPLES IN TABLE I.

Sepal Length ∈			Sepal Width ∈			Petal Length ∈			Petal Width ∈		
$[4.3, 5.5)$	$[5.5, 6.4)$	$[6.4, 7.9]$	$[2.0, 3.0)$	$[3.0, 3.3)$	$[3.3, 4.4]$	$[1.0, 3.0)$	$[3.0, 5.0)$	$[5.0, 6.9]$	$[0.1, 1.0)$	$[1.0, 1.7)$	$[1.7, 2.5]$
1	0	0	0	0	1	1	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	1	0	0	1	0	0	1

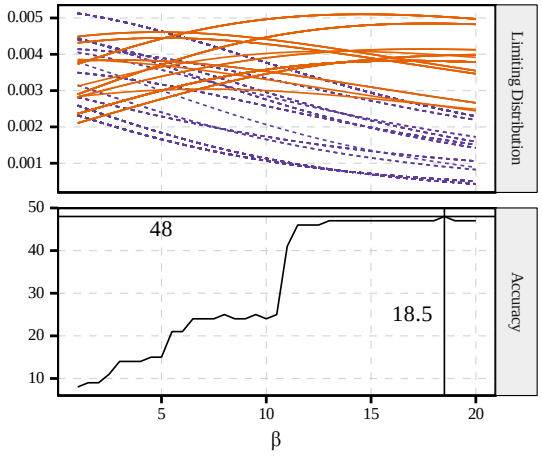


Figure 4. (Top) Limiting distribution of the process modeled from the Iris dataset with two labeled sample of class *Iris virginica* for varying β. Only values associated with unlabeled samples are shown. Solid orange lines and dashed purple lines are associated with positive and negative samples in that order. (Bottom) Accuracy for varying β, which the former is the number of positive samples among the 50 samples with greatest scores.

3) Random Walk: With the network and Equation (3), we compute the transition matrix P. For one labeled sample of species *Iris setosa*, we find the stationary distribution $\vec{\pi}$ iteratively, with $\beta = 6$. Independently of the initial distribution, we should obtain the same results. Figure 3 shows the evolution of $\pi_i(t)$, i such that $\vec{x}_i$ is unlabeled, for two initial distributions $\vec{\pi}(0)$: the uniform distribution, on the left-hand plot, and the limiting distribution of a process P given that $\beta = 1$, that is, ignoring the labeled sample.

The probability of a random walker be in a state of unlabeled positive samples (solid orange lines) surpasses the same on negative samples (dashed purple lines). This behavior holds for both initial distributions and is guaranteed independently of the initial configuration. Starting with the limiting distribution ignoring the labeled sample, the limiting probabilities associated with positive samples clearly increase over time.

We also study the learning process for the second scenario. For $\beta = 1, 1.5, \ldots, 20$, Figure 4 shows the limiting distribution and the accuracy, which is the number of samples of class *Iris virginica* among the 50 samples with greatest scores $f(\vec{x})$. Differently from the former scenario, for small β the probability of a random walker be in a state associated with positive samples (solid orange lines) is not consistently superior to the same on negative samples (dashed purple line). However, for $\beta > 11$, more than 40 out of 50 samples are from the *Iris virginica* class. With $\beta = 18.5$, only two samples are misclassified.

Table III
UCI DATASETS ALONG THE CLASS CHOSEN TO BE THE POSITIVE ONE. P^+ IS THE PROPORTION OF SAMPLES IN THE INDICATED CLASS.

Dataset	Positive class	P^+
Breast 2010	adi	0.21
Ecoli	cp	0.43
Glass	building windows non float processed	0.36
Iris	setosa, versicolor, virginica	0.33
Wine	two	0.40

B. Performance Comparison

We compare our model with a baseline semi-supervised method. Such baseline method is based on the neighborhood graph. Both approaches not only rely on the same assumptions for the input dataset but also use the same classification mechanism; providing a fair comparison of the positive-class pertinence levels.

1) Learning from the Neighborhood Graph: The k-NN graph of a dataset $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ is a graph where each vertex v_i, associated with sample $\vec{x}_i$, connects with all vertices v_j such that $\vec{x}_j$ is within the k-neighborhood of $\vec{x}_i$. We use the asymmetric binary similarity—proportional to the number of attributes that are present in both samples—to calculate the neighborhoods.

The positive-class pertinence level $f^{k\text{-NN}}(\vec{x}_i)$, for all $i \notin \mathcal{P}$, is $(\min_{j\in\mathcal{P}} l_{ij})^{-1}$, where l_{ij} is the length of the shortest path between vertices v_i and v_j in the k-NN graph. In other words, the positive class pertinence of a sample is inversely proportional to the shortest distance from the associated vertex to any labeled vertex.

2) Benchmark Datasets: Five UCI datasets [12] are used to compare our model with the k-NN method. In each dataset, the largest class is the positive one. In the Iris dataset, however, the classes are of the same size, and all three possible cases were considered. Table III presents the datasets along with the proportion P^+ of positive samples; which we use as the positive-class prior probability.

3) Separateness: The *separateness* evaluates how well a model differentiates positive from negative samples. Given the positive pertinence $f(\vec{x}_i)$ for all unlabeled samples $\vec{x}_i$, the separateness is

$$\frac{1}{N - |\mathcal{P}|} \sum_{i \notin \mathcal{P}} \tilde{f}(\vec{x}_i)\delta(\vec{x}_i) \tag{6}$$

where $\tilde{f}$ is the pertinence level of unlabeled samples normalized with standard score, and $\delta(\vec{x}_i)$ is $+1$ if $\vec{x}_i$ is positive or

Table IV
SEPARATENESS OF THE METHODS ON THE UCI DATASETS. FOR EACH SETTING, THE BEST PARAMETER COMBINATION IS SHOWN.

Dataset	Our method	(β, m)	k-NN graph	(k, m)
	1% labeled			
Breast 2010	$\mathbf{0.60 \pm 0.11}$	(10, 5)	0.53 ± 0.13	(20, 4)
Ecoli	$\mathbf{0.60 \pm 0.12}$	(50, 3)	0.56 ± 0.07	(15, 4)
Glass	0.12 ± 0.14	(50, 4)	$\mathbf{0.15 \pm 0.14}$	(15, 4)
Iris (setosa)	$\mathbf{0.91 \pm 0.01}$	(20, 3)	0.75 ± 0.05	(20, 5)
Iris (versicolor)	$\mathbf{0.77 \pm 0.04}$	(20, 3)	0.58 ± 0.08	(20, 3)
Iris (virginica)	$\mathbf{0.65 \pm 0.01}$	(35, 2)	0.53 ± 0.29	(20, 3)
Wine	$\mathbf{0.61 \pm 0.06}$	(5, 5)	0.40 ± 0.19	(7, 4)
	10% labeled			
Breast 2010	$\mathbf{0.65 \pm 0.04}$	(15, 5)	0.54 ± 0.09	(19, 4)
Ecoli	$\mathbf{0.78 \pm 0.02}$	(50, 3)	0.78 ± 0.04	(19, 4)
Glass	0.23 ± 0.08	(50, 3)	$\mathbf{0.25 \pm 0.07}$	(14, 4)
Iris (setosa)	$\mathbf{0.91 \pm 0.00}$	(5, 3)	0.81 ± 0.04	(20, 5)
Iris (versicolor)	$\mathbf{0.81 \pm 0.03}$	(50, 3)	0.68 ± 0.14	(15, 3)
Iris (virginica)	$\mathbf{0.79 \pm 0.03}$	(35, 3)	0.54 ± 0.13	(17, 3)
Wine	$\mathbf{0.77 \pm 0.04}$	(20, 3)	0.55 ± 0.15	(17, 4)

-1 otherwise. The standard score normalization subtracts all pertinence levels f by the average value and divides by the standard deviation.

The separateness is positive when the majority of positive-labeled samples has score above the average. Negative values or near zero indicate failure to distinguish between positive and negative classes.

4) Discrimination Results: To evaluate our technique, the binary representation was created with numerical attributes discretized in $m = 2, 3, 4, 5$ intervals. The parameter k ranges from 1 to 20, and the parameter β in $\{5, 10, \ldots, 50\}$. The number of initially labeled positive samples are 1% or 10% of positive samples.

Our method separates better in seven out of eight settings. The results are independent of the number of labeled samples, which are listed in Table IV along with the k-NN's results. The table shows the mean separateness and the standard deviation for 20 independent sets of labeled items, with the parameters that yield the best performance. The proposed technique present relative low standard deviation, implying the model is less sensible to the choice of labeled samples.

5) Classification Results: Excellent accuracy is observed in our method for few labeled samples and the naive classification mechanism. Using the predicted class c and the same parameter settings, Table V shows the accuracies for the UCI datasets. Surprisingly, the k-NN method obtained better accuracy for the Ecoli dataset besides its worse separateness. Our method's results compare to those acquired by recently proposed techniques that profit from more than twice of labeled samples [13].

C. Document Classification

We also perform tests on the Reuters-21578 ApteMod dataset. This dataset is a collection of 10,788 documents from the Reuters financial newswire service. Each document belongs to one or more of the 90 categories.

Table V
ACCURACY OF THE METHODS ON THE UCI DATASETS. FOR EACH SETTING, THE BEST PARAMETER COMBINATION IS SHOWN.

Dataset	Our method	(β, m)	k-NN graph	(k, m)
	1% labeled			
Breast 2010	$\mathbf{0.91 \pm 0.07}$	(5, 5)	0.88 ± 0.05	(20, 4)
Ecoli	0.87 ± 0.02	(5, 3)	$\mathbf{1.00 \pm 0.00}$	(1, 2)
Glass	$\mathbf{0.58 \pm 0.07}$	(50, 2)	0.57 ± 0.08	(20, 4)
Iris (setosa)	$\mathbf{1.00 \pm 0.00}$	(10, 3)	$\mathbf{1.00 \pm 0.00}$	(1, 2)
Iris (versicolor)	$\mathbf{0.90 \pm 0.03}$	(40, 3)	0.80 ± 0.14	(16, 5)
Iris (virginica)	$\mathbf{0.84 \pm 0.15}$	(30, 3)	0.81 ± 0.18	(20, 3)
Wine	$\mathbf{0.78 \pm 0.03}$	(5, 5)	0.71 ± 0.11	(14, 3)
	10% labeled			
Breast 2010	$\mathbf{0.95 \pm 0.02}$	(15, 5)	0.86 ± 0.05	(12, 5)
Ecoli	0.91 ± 0.02	(10, 3)	$\mathbf{1.00 \pm 0.00}$	(1, 3)
Glass	0.64 ± 0.05	(50, 3)	$\mathbf{0.66 \pm 0.05}$	(13, 4)
Iris (setosa)	$\mathbf{1.00 \pm 0.00}$	(5, 3)	$\mathbf{1.00 \pm 0.00}$	(1, 2)
Iris (versicolor)	$\mathbf{0.93 \pm 0.02}$	(40, 3)	0.91 ± 0.03	(20, 4)
Iris (virginica)	$\mathbf{0.95 \pm 0.03}$	(20, 3)	0.76 ± 0.06	(14, 3)
Wine	$\mathbf{0.91 \pm 0.03}$	(35, 4)	0.79 ± 0.07	(18, 4)

Table VI
AVERAGE F-SCORE ON THE REUTERS DATASET WITH DIFFERENT CLASSES AS POSITIVE. FOR EACH SETTING, THE BEST NUMBER OF SELECTED FEATURES D IS SHOWN. COMPARATIVE RESULTS OBTAINED FROM [14].

Positive class	Our method (D)	EN D	SNOB MC D	TBSVM
Earn	0.431 (28)	0.573	**0.575**	0.536
Acq	0.347 (23)	0.445	**0.495**	0.431
Money-fx	**0.243** (8)	0.188	0.215	0.174
Grain	0.228 (4)	0.190	**0.233**	0.166
Crude	**0.290** (4)	0.191	0.226	0.172
Trade	**0.213** (12)	0.180	0.195	0.162
Interest	**0.249** (5)	0.135	0.152	0.133
Ship	**0.274** (5)	0.116	0.143	0.105
Wheat	**0.441** (3)	0.122	0.144	0.113
Corn	**0.256** (2)	0.097	0.108	0.084
Average	**0.267** (5)	0.224	0.249	0.208

Our method is compared with state-of-the-art algorithms studied in [14]. All methods are adaptations of support vector machines. For each of the top 10 most populated categories, we derive a learning scenario where documents that belong to that category are treated as positive. We label 20% of the positive samples using a biased labeling process, which labels highly correlated documents; refer to [14] for more details.

This learning setting represents real PU learning tasks excellently. The biased labeling process reproduces the most common scenario where the positive examples are labeled based on search queries – for example, searches filtered by keywords – rather than uniformly at random [14].

For each document, we tokenize the text in words and stem each word, obtaining 21,173 different word stems. The binary representation of the dataset is straightforward: each stem is a feature, and a sample document has value 1 for a stem if it occurs in the document. Since there are many features, we pre-select only the D most frequent stems in the set of labeled positive samples. For all experiments, we fix $\beta = 10^5$; which has been found empirically for best results.

Table VI presents the classification results. Each value is the average F-score from 5 independent labeled seeds; except

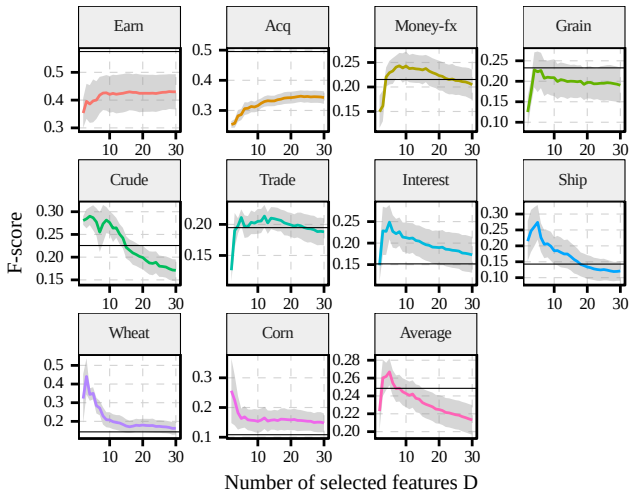


Figure 5. Average F-score and 95% confidence interval obtained by our method over the number of selected features D. The solid horizontal lines indicate the results of the state-of-the-art technique SNOB MC Double [14].

for our method, where we consider 20 independent random seeds. For each topic, we indicate the best number of selected features D.

In Figure 5, one can observe the F-score average and the 95% confidence interval obtained for our method over the number of selected features. The horizontal lines indicate the state-of-the-art results.

Our model has the best results for almost all categories, except for the two most frequent topics. In average, our technique also outperforms the other methods. Both the number of selected features and the prior probabilities play a crucial role in the process. In future works, we shall conduct studies to estimate their optimal values.

V. Discussion

The proposed model is simple to understand, to implement, and efficient to solve PU learning tasks with few or several labeled samples. Moreover, if the input dataset comes in a sparse representation, the computational complexity is linear. Nonetheless, three limitations will be addressed in the future.

Separateness is unsuitable for imbalanced datasets because it normalizes the class pertinence towards the average. A measurement that normalizes towards an estimated prior probabilities might provide better results.

The optimal β is nontrivial, and a lower bound estimative that results in greater limiting probabilities for labeled samples should be possible.

The classification step depends on the assumption of knowing the positive-class prior. Such information may not feature in real datasets.

We can also extend our technique to several other learning problems: *a)* An alternative method for multi-class datasets can be obtained with competition dynamics instead of random walking [5]. *b)* Since addition and removal of samples in the feature–sample network and updates on the eigenvalues and eigenvectors cost little resources, the method can be adapted to deal with concept drift, outlier detection, classification on data streams, and active learning. *c)* An analysis of the limiting probabilities for the states associated with features would lead to new techniques for feature selection and dimensionality reduction.

VI. Conclusion

We presented a PU learning system for transductive classification. We map the input data into a sparse binary representation and, afterward, into a complex network whose vertices represent samples and attributes. From only positive-labeled and unlabeled instances, we model a Markov Chain that outputs positive-class pertinence levels for the unlabeled samples. Knowing the priori probability of the positive class, we classify the unlabeled samples.

The model is illustrated step by step and evaluated against a baseline method and other techniques in literature. The proposed scheme offers high performance on PU learning problems, even with few labeled samples. However, two main limitations can impair its use on real applications: the estimation of the parameter β and the class priors.

Once we address these issues, we can further explore the feature–sample network, generalize and extend the technique to other learning problems.

Acknowledgments

This research was supported by the São Paulo State Research Foundation (FAPESP) and the Brazilian National Research Council (CNPq).

References

[1] X. Zhu and A. B. Goldberg, "Introduction to Semi-Supervised Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

[2] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, 2008.

[3] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling Up Graph-Based Semisupervised Learning via Prototype Vector Machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 444–457, 2015.

[4] T. C. Silva and L. Zhao, *Machine Learning in Complex Networks*, 1st ed. New York, NY, USA: Springer, 2016.

[5] F. A. N. Verri, P. R. Urio, and L. Zhao, "Network Unfolding Map by Edge Dynamics Modeling," 2016, arXiv:1603.01182.

[6] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM," *Machine Learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[7] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data," in *International Joint Conference on Artificial Intelligence Proc.*, 2003, pp. 587–592.

[8] J. Mũnoz Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, "Semisupervised One-Class Support Vector Machines for Classification of Remote Sensing Data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 8, pp. 3188–3197, 2010.

[9] S. Yu and C. Li, "PE-PUC: A Graph Based PU-Learning Approach for Text Classification," in *Machine Learning and Data Mining in Pattern Recognition*. Springer Berlin Heidelberg, 2007, vol. 4571, pp. 574–584.

[10] S. Segui, L. Igual, and J. Vitria, "Weighted Bagging for Graph Based One-Class Classifiers," in *Multiple Classifier Systems Proc.*, 2010, vol. 5997, pp. 1–10.

[11] S. M. Ross, *Introduction to Probability Models*, 11st ed. Orlando, FL, USA: Academic Press, 2014.

[12] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[13] L. Livi, A. Sadeghian, and W. Pedrycz, "Entropic One-Class Classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3187–3200, 2015.
[14] N. Youngs, D. Shasha, and R. Bonneau, "Positive-Unlabeled Learning in the Face of Labeling Bias," in *IEEE International Conference on Data Mining Workshop Proc.*, 2015, pp. 639–645.

CHAPTER

5

FEATURE LEARNING IN FEATURE–SAMPLE NETWORKS USING MULTI-OBJECTIVE OPTIMIZATION

Preprint, arXiv:1710.09300 [cs.AI], <https://arxiv.org/abs/1710.09300>, Filipe Alves Neto Verri, Renato Tinós, Liang Zhao, "Feature learning in feature-sample networks using multi-objective optimization", submitted to the IEEE Congress on Evolutionary Computation, 2018.

Contribution statement

F.A.N. Verri conceived and developed the research. F.A.N. Verri, R. Tinós and L. Zhao wrote the manuscript. R. Tinós and L. Zhao supervised the project.

Feature learning in feature–sample networks using multi-objective optimization

Filipe Alves Neto Verri
Institute of Mathematical and Computer Sciences
University of São Paulo – São Carlos, SP, Brazil
E-mail: filipeneto@usp.br

Renato Tinós, Liang Zhao
Department of Computing and Mathematics, FFCLRP
University of São Paulo – Ribeirão Preto, SP, Brazil
E-mail: rtinos@ffclrp.usp.br, zhao@usp.br

***Abstract*—Data and knowledge representation are fundamental concepts in machine learning. The quality of the representation impacts the performance of the learning model directly. Feature learning transforms or enhances raw data to structures that are effectively exploited by those models. In recent years, several works have been using complex networks for data representation and analysis. However, no feature learning method has been proposed for such category of techniques. Here, we present an unsupervised feature learning mechanism that works on datasets with binary features. First, the dataset is mapped into a feature–sample network. Then, a multi-objective optimization process selects a set of new vertices to produce an enhanced version of the network. The new features depend on a nonlinear function of a combination of preexisting features. Effectively, the process projects the input data into a higher-dimensional space. To solve the optimization problem, we design two metaheuristics based on the lexicographic genetic algorithm and the improved strength Pareto evolutionary algorithm (SPEA2). We show that the enhanced network contains more information and can be exploited to improve the performance of machine learning methods. The advantages and disadvantages of each optimization strategy are discussed.**

***Index Terms*—Feature learning, multi-objective optimization, complex networks, genetic algorithm**

I. Introduction

A good representation of the encoded knowledge in a machine learning model is fundamental to its success. Several data structures have been used for this purpose, for instance, matrices of weights, trees, and graphs [1], [2]. Sometimes, learning models lack a data structure to store knowledge, storing the input as is [3].

In recent years, several works have been using *complex networks* for data representation and analysis [4]–[6]. Complex networks are graphs with a nontrivial topology that represent the interactions of a dynamical system [7]. Advances in the science of complex systems bring several tools to understand such systems.

In [8], we describe how to map a dataset with binary features into a bipartite complex-network. Such network is called *feature–sample network*. Using that representation, we solve a semi-supervised learning task called positive-unlabeled (PU) learning [9].

When dealing with machine learning problems, we often need to pre-process the input data. *Feature learning* transforms or enhances raw data to structures that are effectively exploited by learning models. Autoencoders and manifold learning are examples of feature learning methods [10].

In this paper, we propose a feature learning process to add information in feature–sample networks. In summary, we include to the network a limited number of new vertices based on a non-linear function of the preexisting ones. The set of new vertices is determined by a multi-objective objective problem, in which the goal is to maximize the number of features while maintaining some properties of the original data. Two multi-objective approaches are designed: a lexicographic genetic algorithm (LGA) and an implementation of the improved strength Pareto evolutionary algorithm (SPEA2).

We show that enhanced feature–sample networks improve the performance of learning methods in the major machine learning paradigms: supervised and unsupervised learning. We also expose the pros and cons of each optimization approach.

The rest of this paper is organized as follows. Sections II and III describe how to enhance feature–sample networks as an optimization problem. In Section IV, computer simulations illustrate the optimization process and assess the performance improvements in machine learning tasks. Finally, we conclude this paper in Section V.

II. Enhanced feature–sample networks

In this section, we describe how we enhance a feature–sample network by adding to the network a constrained number of new features. Connections between the samples and each new feature depend on a nonlinear function of a combination of preexisting features. The chosen set of new features is the result of an optimization process that not only maximizes the number of new features but also distributes them along the samples. This process is similar to the projection of the associated data into a higher-dimensional space preserving some important characteristics of the data.

In the following subsections, we first review the feature–sample networks, then describe the creation of new features. Moreover, we elaborate an optimization problem to enhance feature–sample networks.

A. Feature–sample network review

Assume as input the dataset $\mathcal{B} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ whose members are binary feature vectors $\vec{x}_i = [x_{i1}, \ldots, x_{iD}] \in \{0,1\}^D$. The feature vectors are sparse, that is, the number of elements with value 1 is much lower than the dimension D.

The *feature–sample network* $\mathcal{G}$ is the bipartite complex-network whose edges connect samples and features of

the dataset $\mathcal{B}$. A simple, unweighted, undirected graph $(\mathcal{V}, \mathcal{E})$ represents such network. The vertex set $\mathcal{V}$ is $\{v_1, \ldots, v_N, v_{N+1}, \ldots, v_{N+D}\}$ and an edge exists between *sample* v_i and *feature* v_{N+j} if $x_{ij} = 1$.

B. And-features definition

According to the *Cover's Theorem* [3], given a not-densely populated space of a classification problem, the chance of it being linearly separable is increased as one cast it in a high-dimensional space nonlinearly.

Since we assume the input feature–sample network is sparse, we can synthesize features nonlinearly to exploit the properties of this theorem. One way to produce new features is using the *and* operator, which is a nonlinear Boolean function.

We call *and-feature* the feature v that links to all samples connected to a given set of two or more preexisting features.

Given a feature–sample network $\mathcal{G}$ with N samples and D features, we can produce an and-feature v for each combination $\mathcal{W}$ of q features such that $|\mathcal{W}| \geq 2$ and $\mathcal{W} \subseteq \{v_{N+1}, \ldots, v_{N+D}\}$. We call q the order of the and-feature v. Thus, the number of possible and-features is

$$\sum_{q=2}^{D} \binom{D}{q} = \sum_{q=2}^{D} \frac{D!}{q!\,(D-q)!} = 2^D - D - 1.$$

In the rest of this paper, we index every possible and-feature using the parameter $\vec{a} = [a_1, \ldots, a_D] \in \{0,1\}^D$ such that $\sum_j a_j \geq 2$. Each element a_j indicates a feature in the network. The feature v_{N+j} is part of the combination if, and only if, $a_j = 1$. Thus, the set $\mathcal{W}$ is $\{v_{N+j} \mid a_j = 1\}$.

Using this notation, we say that the and-feature $v(\vec{a})$ connects to each sample v_i if, and only if, $(\neg a_1 \vee x_{i1}) \wedge \cdots \wedge (\neg a_D \vee x_{iD}) = 1$ holds.

From this discussion, we realize that enumerating every combination has exponential cost. Moreover, once the network is sparse, we expect that many of the and-features have no connections.

C. Optimization problem definition

The problem of enhancing the network can be viewed as a optimization problem. Given an input feature–sample network $\mathcal{G}$ with N samples and D features, we denote $\mathcal{G}(\mathcal{Y})$ the enhanced network from the original $\mathcal{G}$ by adding every and-feature $v \in \mathcal{Y}$. The number of features of the enhanced network, excluding the and-features that do not connect, is given by $D(\mathcal{Y})$. Thus, $\mathcal{G} = \mathcal{G}(\varnothing)$ and $D = D(\varnothing)$.

Let $M_{\max}$ be the maximum number allowed of generated features, we want to

$$\begin{aligned} \underset{\mathcal{Y}}{\text{minimize}} \quad & D(\varnothing) - D(\mathcal{Y}) \\ \text{subject to} \quad & D(\mathcal{Y}) - D(\varnothing) \leq M_{\max}. \end{aligned}$$

The disadvantage of this approach is that the and-features might not be well distributed. Thus, while some samples may have few new feature, others may have many. To overcome this limitation, we introduce the *disproportion* $\Delta(\mathcal{G}, \mathcal{G}') \in [0, \infty)$ between the network $\mathcal{G}$ and its enhanced version $\mathcal{G}'$.

The disproportion is zero if the number of new connections in each sample is proportional to its initial sparsity. In this way, while the sparsity of each sample might change, we keep the same shape of the degree distribution of the samples.

Let k_i be the degree of the sample v_i, the disproportion between two networks is

$$\Delta(\mathcal{G}, \mathcal{G}') = \operatorname{sd}\left(\frac{k_1(\mathcal{G}') - k_1(\mathcal{G})}{k_1(\mathcal{G})}, \ldots, \frac{k_N(\mathcal{G}') - k_N(\mathcal{G})}{k_N(\mathcal{G})} \right),$$

where sd is the standard deviation of the arguments.

Using the disproportion in our optimization problem, the goal is to

$$\begin{aligned} \underset{\mathcal{Y}}{\text{minimize}} \quad & D(\varnothing) - D(\mathcal{Y}),\ \Delta\big(\mathcal{G}(\varnothing), \mathcal{G}(\mathcal{Y})\big) \\ \text{subject to} \quad & D(\mathcal{Y}) - D(\varnothing) \leq M_{\max}. \end{aligned}$$

III. Methods

In this section, we study the multi-objective problem stated in the previous section and describe how we solve it.

A. Problem study

In Section II-B, we see that the number of possible and-features scales exponentially in the number of features D. As a consequence, storing every possible and-feature is not feasible.

Furthermore, the number of candidate solutions is also exponential in the number of possible new features. Precisely, there are at the most

$$\sum_{m=1}^{2^D - D - 1} \binom{2^D - D - 1}{m} = \sum_{m=1}^{2^D - D - 1} \frac{\big(2^D - D - 1\big)!}{m!\,(2^D - D - m - 1)!}$$

solutions $\mathcal{Y}$ to explore. We can not limit the size of the set $\mathcal{Y}$ by $M_{\max}$ since many and-features may have no connection.

The three common approaches for solving multi-objective optimization problems are weighted-formula, lexicographic, and Pareto [11]. The first strategy transforms the problem into a single-objective one, usually by weighting each objective and adding them up. The lexicographic approach assigns a priority to each objective and then optimizes the objectives in that order. Hence, when comparing two candidate solutions, the highest-priority objective is compared and, if equivalent, the second objective is compared. If the second objective is also equivalent between the solutions, the third one is used, and so on. Both the weighted-formula and the lexicographic strategies return only one solution for the problem. The Pareto methods use different mathematical tools to evaluate the candidate solution, finding a set of non-dominated solutions. A solution is said to be non-dominated if it is not worse than any other solution concerning all the criteria [11].

Since it is not trivial the difference of scale between our objective functions, we opted to use only a lexicographic and a Pareto method.

We design two population-based optimization algorithms. Specifically, we consider the use of two metaheuristics:

- a lexicographic genetic algorithm (GA); and
- the improved strength Pareto evolutionary algorithm (SPEA2) [12].

Although the methods are different, both approaches share many properties – individual representation, population initialization, operators of mutation, recombination and selection – which are explained in Section III-D. The main difference between them is the fitness evaluation.

In the GA, the individuals are ordered lexicographically, that is, ordered primarily by the first objective function and, in case of tie, the second objective. SPEA2, however, consider not only the Pareto front but also the density of the solutions.

B. Lexicographic genetic algorithm review

A GA has the following steps [13]:

1: $X \leftarrow$ INITIALPOPULATION()
2: **while** STOPCONDITION() = false **do**
3: EVALUATE(X)
4: $X_{\text{next}} \leftarrow$ ELITISM(X)
5: **while** $|X_{\text{next}}| < |X|$ **do**
6: parents $\leftarrow$ SELECT(X)
7: children $\leftarrow$ RECOMBINE(parents)
8: MUTATE(children)
9: $X_{\text{next}} \leftarrow X_{\text{next}} \cup$ children
10: **end while**
11: $X \leftarrow X_{\text{next}}$
12: **end while**.

In words, a random population of candidate solutions is generated as the first step. Then, while the stop condition is not met, the next generation of individuals comprises the best individuals of the previous generation and the individuals originated by recombining and mutating parents selected from the previous generation.

In the lexicographic approach, the only difference is during the evaluation of the candidate solution [14], where the best individuals are decided lexicographically.

In our specific problem, a solution $\mathcal{Y}_1$ is better than $\mathcal{Y}_2$ if

- $D(\mathcal{Y}_1) > D(\mathcal{Y}_1)$; or
- $D(\mathcal{Y}_1) = D(\mathcal{Y}_1)$ and $\Delta\big(\mathcal{G}, \mathcal{G}(\mathcal{Y}_1)\big) < \Delta\big(\mathcal{G}, \mathcal{G}(\mathcal{Y}_1)\big)$.

C. Improved Strength Pareto Evolutionary Algorithm review

SPEA2 works similarly to a GA. The major difference is that it keeps an archive with the candidate solutions for the Pareto set. If the number of non-dominated solutions is greater than the limit of the archive, some solutions are discarded. Such operation is called truncation. The truncation operator tries to maintain the candidate solutions uniformly distributed along the Pareto front [12].

As indicated by the authors, we select individuals by employing binary tournament. Also, let A be the archive size, we fix the parameter $k = \sqrt{\mathrm{A}}$ in the truncation operator [12].

D. Metaheuristic design

The common implementation characteristics of our metaheuristic is exposed in the next items.

1) Individual representation: In our problem, each solution is a set $\mathcal{Y}$ of zero or more and-features. If we enumerate every possible and-feature, the solution can also be viewed as a binary vector with entries 1 for the present and-features.

2) Population initialization: Given $\mu, \sigma > 0$, we sample, without replacement, $\lfloor M \rfloor$ random and-features to compose each candidate solution $\mathcal{Y}$ such that $M \sim \mathcal{N}(\mu, \sigma)$. And-features are sampled so that the probability of having order $q \geq 2$ is

$$\frac{q-1}{q!}.$$

3) Recombination operator: We use the uniform crossover operator with two parents generating two children. In Section III-D6, we show how to implement it efficiently using our set representation.

4) Selection operator: The binary tournament method is chosen to select the parent that go to the recombination step.

5) Mutation operator: We formulated the following specific mutation operator to exploit the characteristics of our problem.

Given a candidate solution $\mathcal{Y}$, we apply $\eta \in \{0, 1, 2, \dots\}$ random changes in the individual. For each change, there is a equal probability of either

- trying to add a new and-feature; or
- trying to remove an and-feature $v \in \mathcal{Y}$; or
- trying to modify an and-feature $v \in \mathcal{Y}$.

When trying to add a new feature, an and-feature v is sampled and the solution is updated to $\mathcal{Y} \cup \{v\}$. Note that the candidate solution $\mathcal{Y}$ may not change if the and-feature was previously present.

If trying to remove a feature, each and-feature $v \in \mathcal{Y}$ has probability $(|\mathcal{Y}| + 1)^{-1}$ of being selected. In this case, the candidate solution is updated to $\mathcal{Y} \setminus \{v\}$. Note that, with probability $(|\mathcal{Y}| + 1)^{-1}$, the individual do not change.

Finally, in the last case, an and-feature $v(\vec{a}) \in \mathcal{Y}$ with order $q = \sum_j a_j$ is selected uniformly to be modified. Once the and-feature is selected, a modified and-feature $v'(\vec{a}')$ will be produced. Two cases may happen: a) with chance $\frac{1}{q}$, an index $j' \in [1, D]$ is selected uniformly, and $\vec{a}'$ is

$$\begin{cases} a'_{j'} = 1 \\ a'_j = a_j \quad \forall j \neq j'; \end{cases}$$

b) with chance $\frac{q-1}{q}$, two indexes $j' \in \{j \mid a_j = 1\}$ and $j'' \in [1, D]$ are chosen uniformly. The modified and-feature is

$$\begin{cases} a'_{j''} = a_{j'} \\ a'_{j'} = a_{j''} \\ a'_j = a_j \qquad \forall j \notin \{j', j''\} \end{cases}$$

The first case will include one more term into the and-feature if $a_{j'} = 0$. The second one swaps two elements of $\vec{a}$ and, effectively, takes effect when $a_{j''} = 0$. The candidate solutions is then updated to $(\mathcal{Y} \setminus \{v\}) \cup \{v'\}$. The size $|\mathcal{Y}|$ is never increased, and the candidate solution will be preserved if $v = v'$.

6) Performance considerations: Although, we can view both solutions and and-features as binary vectors, the set representation is more practical because of the high space-complexity of the problem. Moreover, there is no need to store

entries for and-features that lack connections. Instead of just ignoring them, we exploit the evaluation step to determine which and-features are useless and remove them from the set.

Also, using the set representation, the crossover of the candidate solutions $\mathcal{Y}_1^{\text{parent}}$ and $\mathcal{Y}_2^{\text{parent}}$ can be implemented efficiently with steps

1: $\mathcal{Y}_1^{\text{child}}, \mathcal{Y}_2^{\text{child}} \leftarrow \mathcal{Y}_1^{\text{parent}} \cup \mathcal{Y}_2^{\text{parent}}$
2: **for** $v \in \mathcal{Y}_1^{\text{parent}} \triangle \mathcal{Y}_2^{\text{parent}}$ **do**
3: **if** SAMPLEUNIFORM(0, 1) < 0.5 **then**
4: $\mathcal{Y}_1^{\text{child}} \leftarrow \mathcal{Y}_1^{\text{child}} \cup \{v\}$
5: **else**
6: $\mathcal{Y}_2^{\text{child}} \leftarrow \mathcal{Y}_2^{\text{child}} \cup \{v\}$
7: **end if**
8: **end for**

where $\triangle$ stands for the symmetric difference operator.

Finally, to conform to the requirements in Section III-D2, one can sample each candidate solution $\mathcal{Y}$ efficiently with steps

1: $v \leftarrow$ SAMPLEANDFEATURE()
2: $\mathcal{Y} \leftarrow \{v\}$
3: **loop**
4: $v \leftarrow$ SAMPLEANDFEATURE()
5: $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{v\}$
6: **if** SAMPLEUNIFORM(0, 1) $< 1 - \frac{1}{|\mathcal{Y}|}$ **then**
7: **break**
8: **end if**
9: **end loop**

where and-features are sampled without replacement.

IV. Experimental results

In this section, we present applications of our feature learning technique in the two major categories in machine learning: supervised and unsupervised learning.

First, we illustrate the optimization process in the famous Iris dataset. In this example, we also conduct community detection in the original feature–sample network obtained from the dataset and in the enhanced one.

Then, we observe the increase of the accuracy obtained by the k-nearest neighbors [1] classifier in other four UCI datasets.

A. Enhanced community detection and clustering

The UCI Iris dataset [15] contains 150 samples that describe the sepals and petals of individual Iris flowers. The flowers are either *Iris setosa*, *Iris virginica*, or *Iris versicolor*. In [8], we construct a feature–sample network from this dataset by discretizing the features.

Figure 1 shows the generated network. Each color represents a different class. Circles represent samples and squares, features. We use that same network as an input to both algorithms – using SPEA2 and the lexicographic GA.

1) Evolution of the candidate solutions: We execute the optimization process once for each strategy. We fix the population in 1000 individuals. For SPEA2, the archive has size 100 and, for the lexicographic GA, we keep the 100 best solutions over the generations. In the initial population, we use $\mu = 10$ and $\sigma = 5$. The recombination rate is 0.6 and the apply $\eta = 1$ random changes in each generated individual.

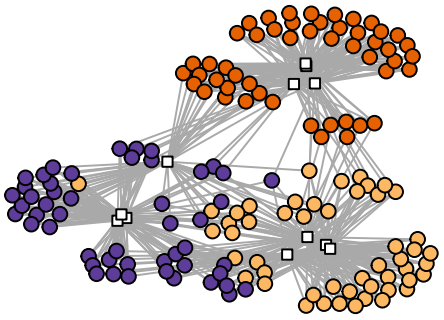


Figure 1. Feature–sample network for Iris dataset. Circles are vertices associated with samples and squares with features. Colors represent the classes.

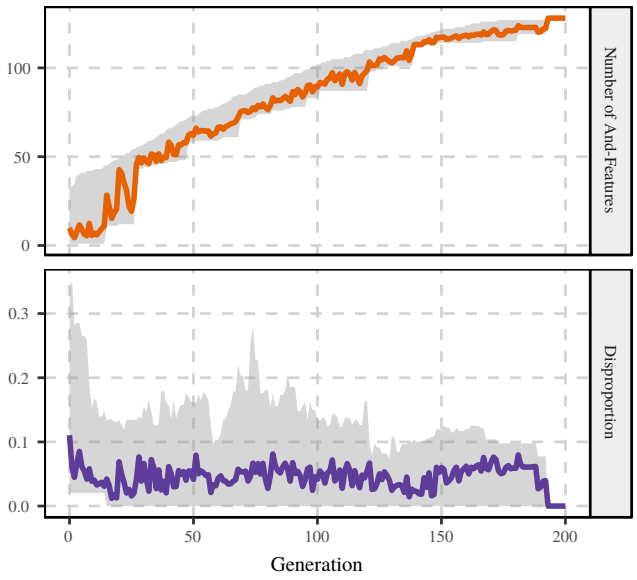


Figure 2. Evolution of the number of and-features and disproportion along the generations of the SPEA2 algorithm with Iris dataset as the input network. Solid lines are the average disproportion and number of and-features of the non-dominated solution at a given generation. Shadows cover the range of the measurements.

Figures 2 and 3 describe the obtained result. Both disproportion and number of discovered and-feature are depicted along the generations. Solid lines are the average result in the population and shadows cover the range – from minimum to maximum – of each measurement. The results include only the non-dominated solutions in SPEA2 and the 100 best solutions in the lexicographic GA.

Using both strategies, we could reach the optimal solution: 128 and-features with at least one connection and 0 disproportion. However, the optimization strategies differ as to how to achieve this.

SPEA2 tries to find as many new and-feature while keeping the ones with lowest values of disproportion. When a larger set of and-features with disproportion 0 is discovered, such solution dominates every other solution found so far. Thus,

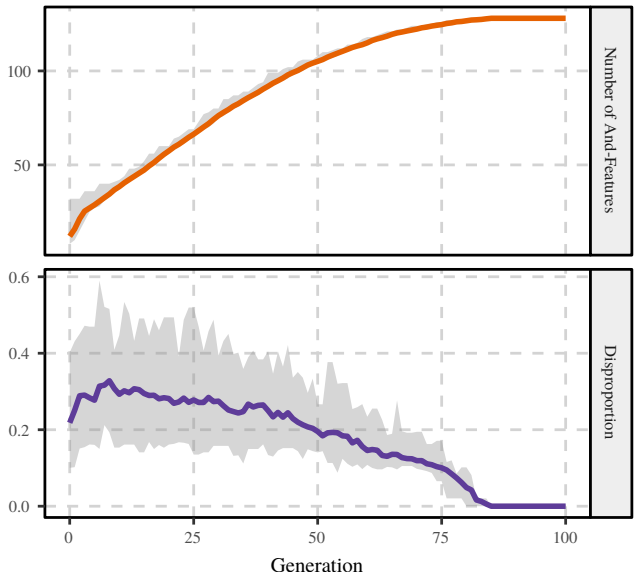


Figure 3. Evolution of the number of and-features and disproportion along the generations of the lexicographic GA with Iris dataset as the input network. Solid lines are the average disproportion and number of and-features of the best 100 solutions (elitism) at a given generation. Shadows cover the range of the measurements.

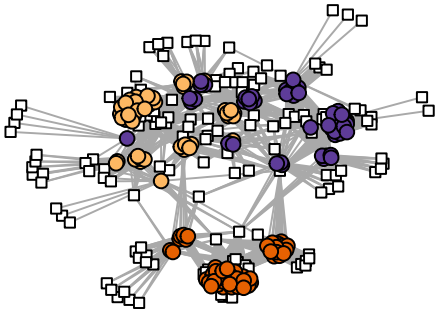


Figure 4. Feature–sample network for Iris dataset with all possible 128 and-features (excluding those with no connections). Circles are vertices associated with samples and squares with features. Colors represent the classes.

we can observe "steps" in the evolution of the number of and-features.

In the lexicographic GA, the disproportion is only considered when the number of discovered and-features is the same. As a result, the algorithm greedily produce and-features disregarding the disproportion until it cannot find more new features to add. It enables a faster convergence in this case, but it may find only solutions with high disproportion when it is unfeasible to reach the maximum number of and-feature – which is very common in practice. To solve this issue in larger problems, one can set the limit in the number of discovered features, $M_{\max}$.

The optimal enhanced feature–sample network for this dataset is in Figure 4.

2) Community detection: Applying a greedy community detection method [16] in both networks, we observe that the enhanced network has modularity 12.4% ($Q = 0.561$) higher than the input network ($Q = 0.499$.)

The enhanced network can also improve clustering tasks. If comparing the expected class and the obtained communities, the enhanced version achieves higher Jaccard index, 0.731 against 0.719.

Table I
UCI DATASETS ALONG WITH THE NUMBER OF SAMPLES N, FEATURES D, AND POSSIBLE AND-FEATURE M.

Dataset	N	D	M
Breast 2010	106	27	134,217,700
Ecoli	336	19	524,268
Glass	214	25	33,554,406
Wine	178	39	549,755,813,848

Table II
NUMBER OF AND-FEATURES AND DISPROPORTION OBTAINED BY THE OPTIMIZATION PROCESS FOR BOTH STRATEGIES. AVERAGE AND STANDARD DEVIATION ARE SHOWN FOR EACH MEASUREMENT.

Dataset	LGA	SPEA2
Breast 2010	2700 ± 0, 0.2 ± 0.03	513.3 ± 272.4, 0.02 ± 0.03
Ecoli	1900 ± 0, 0.13 ± 0.01	372.6 ± 268.3, 0.04 ± 0.04
Glass	2500 ± 0, 0.14 ± 0.01	481 ± 399, 0.05 ± 0.04
Wine	3900 ± 0, 0.28 ± 0.03	561 ± 452.8, 0.03 ± 0.02

B. Performance enhancement in supervised learning

We also apply our proposal in 4 classification tasks from UCI [15]. Table I presents the datasets along with the number of samples N, features D, and possible and-features M. We highlight that it is unfeasible to list every possible combination among the features even for small datasets. The networks are generated as shown in [8] with 3 bins.

The optimization process is executed 15 times for each strategy – SPEA2 and GA. We fix the population size in 1000, the archive size in 100, and the elitism 100 solutions. For the initial population, we use $\mu = 50$ and $\sigma = 10$. The recombination rate is 0.6 and $\eta = 1$ mutations are performed for each candidate solution. We limit the number of and-feature by $M_{max} = 100D$. The execution is stopped after the 1000th generation.

Table II summarizes the number of and-features and disproportion obtained by the optimization process. For the lexicographic GA, we show the average and the standard deviation of the measurements among the 100 best individuals. For SPEA2, only the non-dominated solutions are considered.

As expected by considering the previous study (Section IV-A,) the lexicographic GA achieved better count of and-features – the maximum allowed –, but worse values of disproportion. The candidate solutions of SPEA2 present wide variation, but consistent lower disproportion.

For each one of the datasets, we take the candidate solution with highest count of and-features and with the lowest disproportion among every solution produced. (We ignored a few solutions with less than 100 and-features.) Such candidate solutions, and the strategy that has found them, are indicated in Table III.

Table III
NUMBER OF AND-FEATURES, DISPROPORTION, AND STRATEGY OF THE RESULTS WITH LOWEST DISPROPORTION AND HIGHEST NUMBER OF AND-FEATURES.

Dataset	Lowest disproportion	Highest number of and-features
Breast 2010	513, 0.000 (SPEA2)	2700, 0.129 (LGA)
Ecoli	166, 0.011 (SPEA2)	1900, 0.114 (LGA)
Glass	416, 0.019 (SPEA2)	2500, 0.130 (LGA)
Wine	413, 0.015 (SPEA2)	3900, 0.226 (LGA)

We solve the classification problems for each one of the datasets using the k-nearest neighbors method. As inputs we use the interaction matrices $(x_{ij} \in \{0,1\})_{ij}$ of the original network and the enhance ones from the selected candidate solutions. We performed 20 split validations with 70 and 80% of labeled samples for each case. We also varied $k \in \{1, 2, \ldots, 20\}$.

The best results for each configuration are shown in Table IV. Improvements are highlighted in bold. Using the solution with the highest number of and-features, we improved the classification results in all cases. Using less and-features but with lower disproportion, we see improvements almost as good as using higher and-feature count.

V. CONCLUSION

In this paper, we presented an unsupervised feature learning mechanism that works on datasets with binary features. First, the dataset is mapped into a feature–sample network. Then, a multi-objective optimization process selects a set of new vertices that correspond to new features to produce an enhanced version of the network.

We show that the enhanced network contains more information and can be exploited to improve the performance of machine learning methods.

To solve the optimization problem, we opted to design population-based metaheuristics. We used both a lexicographic GA and the SPEA2 algorithm to find the candidate solutions.

From the experiments, we conclude that the GA produces more new features in fewer generations. However, candidate solutions in SPEA2, besides having less new features, also improved the performance of machine learning methods.

This fact suggests that the disproportion is a good measurement of the quality of the selected set of and-features. In future works, we will correlate improvement and disproportion of the solutions with the same number of features.

Furthermore, the learning techniques used – fast-greedy community detection and k-nearest neighbors – do not take full advantage of the new features. In subsequent studies, we will elaborate learning models to exploit the enhanced feature–sample network explicitly.

ACKNOWLEDGMENTS

This research was supported by the São Paulo State Research Foundation (FAPESP) and the Brazilian National Research Council (CNPq).

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
[2] X. Zhu and A. B. Goldberg, "Introduction to Semi-Supervised Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.
[3] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 3, pp. 326–334, 1965.
[4] T. C. Silva, L. Zhao, and T. H. Cupertino, "Handwritten data clustering using agents competition in networks," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 3, pp. 264–276, 2013.
[5] T. C. Silva and L. Zhao, *Machine Learning in Complex Networks*, 1st ed. Springer, 2016.
[6] F. A. N. Verri, P. R. Urio, and L. Zhao, "Network unfolding map by vertex-edge dynamics modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2016.
[7] M. E. J. Newman, *Networks: An Introduction*, 1st ed. New York, NY: Oxford University Press, 2010.
[8] F. A. N. Verri and L. Zhao, "Random walk in feature-sample networks for semi-supervised classification," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, 2016, pp. 235–240.
[9] J. Mũnoz Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, "Semisupervised One-Class Support Vector Machines for Classification of Remote Sensing Data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 8, pp. 3188–3197, 2010.
[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[11] A. A. Freitas, "A critical review of multi-objective optimization in data mining: A position paper," *SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 77–86, 2004.
[12] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Athens, Greece: International Center for Numerical Methods in Engineering, 2001, pp. 95–100.
[13] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.
[14] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.
[15] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[16] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, pp. 1–6, 2004.

Table IV
ACCURACY OF THE k-NN METHOD ON THE UCI DATASETS USING THREE DIFFERENT INPUT NETWORKS – ORIGINAL, LOWEST DISPROPORTION, AND HIGHEST NUMBER OF AND-FEATURES. FOR EACH SETTING, THE BEST VALUE OF k IS SHOWN.

Dataset	Original	(k)	Lowest disproportion	(k)	Highest number of and-features	(k)
			70% labeled			
Breast 2010	0.62 ± 0.07	(1)	0.63 ± 0.07	(1)	$\mathbf{0.64 \pm 0.06}$	(1)
Ecoli	0.76 ± 0.03	(4)	$\mathbf{0.77 \pm 0.03}$	(4)	0.76 ± 0.04	(12)
Glass	0.66 ± 0.06	(7)	0.69 ± 0.05	(6)	$\mathbf{0.71 \pm 0.05}$	(8)
Wine	0.92 ± 0.03	(17)	$\mathbf{0.93 \pm 0.03}$	(3)	$\mathbf{0.93 \pm 0.02}$	(5)
			80% labeled			
Breast 2010	0.65 ± 0.11	(5)	0.62 ± 0.10	(4)	$\mathbf{0.66 \pm 0.10}$	(3)
Ecoli	0.77 ± 0.03	(6)	$\mathbf{0.78 \pm 0.04}$	(4)	$\mathbf{0.78 \pm 0.04}$	(13)
Glass	0.66 ± 0.07	(7)	0.69 ± 0.06	(6)	$\mathbf{0.72 \pm 0.07}$	(9)
Wine	0.93 ± 0.05	(19)	$\mathbf{0.94 \pm 0.05}$	(3)	$\mathbf{0.94 \pm 0.03}$	(3)

CHAPTER

6

NETWORK COMMUNITY DETECTION VIA ITERATIVE EDGE REMOVAL IN A FLOCKING-LIKE SYSTEM

Preprint, arXiv:1802.04186 [cs.SI], <https://arxiv.org/abs/1802.04186>, Filipe Alves Neto Verri, Roberto Alves Gueleri, Qiusheng Zheng, Junbao Zhang, Liang Zhao, "Network community detection via iterative edge removal in a flocking-like system", submitted to IEEE Transactions on Evolutionary Computation, 2018. (This version includes some improvements.)

Contribution statement

R.A. Gueleri, Q. Zheng, J. Zhang, and L. Zhao conceived the community detection model. R.A. Gueleri performed the experiments that resulted in Figures 1, 4, 5, 6, 7, 8, and 9. F.A.N. Verri conducted the experiments that resulted in Figures 2, 3, and 10, and Table 1. F.A.N. Verri devised the analytical results. All authors contributed on writing the manuscript. L. Zhao, in addition, supervised all this research work.

Network community detection via iterative edge removal in a flocking-like system

Filipe Alves Neto Verri, Roberto Alves Gueleri, Qiusheng Zheng, Junbao Zhang, Liang Zhao

***Abstract*—We present a network community-detection technique based on properties that emerge from a nature-inspired system of aligning particles. Initially, each vertex is assigned a random-direction unit vector. A nonlinear dynamic law is established so that neighboring vertices try to become aligned with each other. After some time, the system stops and edges that connect the least-aligned pairs of vertices are removed. Then the evolution starts over without the removed edges, and after enough number of removal rounds, each community becomes a connected component. The proposed approach is evaluated using widely-accepted benchmarks and real-world networks. Experimental results reveal that the method is robust and excels on a wide variety of networks. Moreover, for large sparse networks, the edge-removal process runs in quasilinear time, which enables application in large-scale networks.**

***Index Terms*—Community detection, modularity optimization, flocking formation, complex networks.**

I. Introduction

THE study of complex networks attracts many researches from different areas. Networks are graphs that represent the relationships among individuals in many real-world complex systems. Each vertex is an object of study, and an edge exists if its endpoints interact somehow [1], [2].

A community structure is commonly found in many real networks, such as social networks [3], [4], oil-water flow structure [5], human mobility networks [6], spatial structure of urban movement in large cities [7], corporate elite networks in the fields of politics and economy [8], and many more. Formally, communities are groups of densely connected vertices [9], [10], [11], while connections between different communities are sparser.

The problem of community detection is related to the graph partition problem in graph theory. Finding the optimal partition is an NP-hard problem in most cases [12], thus making room for many researches to find out sub-optimal solutions in feasible time. As a result, various approaches for community detection have been developed, including spectral properties of graph matrices [13], [14], particle walking and competition in networks [1], [11], and many evolutionary or bio-inspired processes [2].

Newman and Girvan [15] have proposed a metric called *modularity*, whose purpose is to quantify how a network is likely to display community structure [16]. It does not make any assumption on *a-priori* knowledge of the network, e.g., vertex labels. One of the algorithms employed in this paper, both for comparison and for complementary stage, is called "Cluster Fast Greed", or just CFG, and it is based on a greedy optimization of modularity [9], [17]. It performs really fast, in time $O(md \log n)$, where d is the depth of the dendrogram describing the network's hierarchical community structure returned by the algorithm. In cases where $m \sim n$ and $d \sim \log n$, it runs in quasilinear time: $O(n \log^2 n)$. Another algorithm employed here is the so-called "Louvain", which is based on the optimization of modularity too [17], [18]. The authors advocate in favour of the computation time, which makes the algorithm applicable on huge networks.

In this paper, we propose a bio-inspired community-detection method that is divided in two alternating stages. The first one is a nonlinear collective complex system that takes inspiration from the flocking formation in nature [19]. In the second stage, we measure the misalignment of each pair of vertices that are directly connected and remove a fraction of edges that result the highest misalignments. Flocks are groups of individuals that move in a coordinated fashion. This coordinated motion emerges even in the absence of any leader, what makes it a self-organizing phenomenon. In our model, each vertex is an aligning particle. Therefore, each vertex carries a velocity vector, pointing to a random direction at the beginning and, as the process evolves, progressively turns itself toward the same direction of its neighboring vertices. As a simplification of the process, the vertices actually do not move, so the term "velocity" is just an analogy to the direction of motion in flocking systems. The dynamical process is suspended after a certain number of iterations, then the second stage takes place. The edge that connects the least aligned pair of vertices is supposed to link distinct communities. After enough number of removal cycles, most of the inter-community edges are expected to be removed, thus the network becomes partitioned into disconnected components.

Our model is evaluated using widely-accepted benchmarks and real-world networks. It not only excels on many different scenarios but also has very low computational cost. As a result, the research shows potential for a broad range of applications, including big data.

The rest of this paper is organized as follows. Sections II and III describe the proposed model and present analytical results of the system. In Section IV, computer simulations illustrate the process and assess its performance. Finally, Section V discusses and concludes this paper.

F.A.N. Verri and R.A. Gueleri are with the Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil.

Q. Zheng and J. Zhang are with the School of Computer Science, Zhongyuan University of Technology, Zhengzhou, China.

L. Zhao is with Ribeirão Preto School of Philosophy, Science and Literature, University of São Paulo, Ribeirão Preto, Brazil.

E-mail: zhao@usp.br

II. Model description

Here we present the iterative edge-removal approach in detail. As mentioned before, the process is divided in two alternating stages: the particle-alignment dynamical process and the edge-removal stage itself.

A. The particle-alignment dynamical model

Given an undirected network, free of self-loops and multiple edges, we let each vertex $i \in \{1, 2, \ldots, N\}$ be a particle in the collective dynamical system. Therefore, each vertex carries a velocity vector $\mathbf{v}_i(t) \in \mathbb{R}^D$, which points to some direction in a multi-dimensional space. The vertices actually do not move at all, so the term "velocity" is just an analogy to the direction of motion in flocking systems, whose moving particles try to align themselves to their neighbors, i.e., try to take the same direction of motion.

Edges of the network define the neighborhood of each particle i, i.e., those particles to which i can directly interact. We denote $\gamma_{ij} = 1$ if vertices i and j interact, and $\gamma_{ij} = 0$ otherwise. The neighborhood is unchangeable throughout the dynamical process, thus the interaction network is the same all the time.

Initially, each particle i is assigned a random initial velocity $\mathbf{v}_i(0)$, which is a unit vector pointing to a random direction. A three-dimensional space ($D = 3$) is employed for all the experiments presented in this paper. However, how dimensionality impact on the final results has to be investigated.

The nonlinear dynamical system is governed by the expression

$$\vec{v}_i(t+1) = \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \alpha \frac{1}{k_i} \sum_{j=1}^{N} \gamma_{ij} \left(\frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} - \frac{\vec{v}_i(j)}{\|\vec{v}_i(j)\|} \right), \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean norm. The nonlinearity of the dynamical system is introduced by the velocity normalization. Parameter t is the iteration index (time), which starts from zero. Parameter $\alpha > 0$ defines how fast the velocities are updated. The value k_i is the degree of vertex i, i.e., the number of neighbors it has,

$$k_i = \sum_{j=1}^{N} \gamma_{ij}. \quad (2)$$

Although the particles do not move – there is even no position defined to them –, by using the unit vector we enforce something analogous to the "constant-speed motion" that each particle would perform. In addition, it enforces that at any given time-step t, all the particles would move at the same speed. Constant-speed motion is a property commonly modelled in studies of self-propelled particles [19], [20]. It is also responsible for the rotational symmetry breaking that makes the set of particles agree on the same velocity direction. By not enforcing such normalization, all velocities would vanish to zero – or close to zero – due to their random initial distribution, making opposite vectors neutralize each other. We show in Section III that particles in the same connected component are most likely to align. As a result, velocity vectors of particles from different communities will converge to the same value. However, as presented in Section IV-A, particles in the same community tend to align from different direction, which motivates the removal of edges some time before the convergence.

B. The iterative edge-removal process

We define the *misalignment coefficient* $H_{ij}(t)$ as the level of disorder in terms of velocity-vectors' misalignment between nodes i and j. Such index is mathematically expressed by

$$H_{ij}(t) = d_1 \left(\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|}, \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right) \quad (3)$$

where $d_1(\cdot, \cdot)$ is the L_1 distance between vectors. In other words, $H_{ij}(t)$ is just the city-block distance between the normalized velocity vectors of the vertices i and j.

As we will see through the experiments presented in the next section, misalignment coefficients of edges that connect distinct communities decrease slower than those connecting vertices inside the same community. It builds the basis of our iterative edge-removal approach: removing the edges with highest misalignment coefficients is likely to remove inter-community edges, thus making the distinction between different communities clearer and clearer over removal cycles. So the overall edge-removal process consists of the following steps:

1) After assigning random initial velocity vectors to every vertex, run the dynamical particle-aligning model, as described in the previous section, up to some number of time-steps. (Number of steps is discussed in Section IV-B.)
2) Once the dynamics is interrupted, collect the misalignment coefficient of each pair of interacting vertices. It is possible to run the dynamical model (Step 1) multiple times, using different random assignments to the set of initial velocities. In this case, the final coefficient of each edge can be just the summation of individual coefficients collected after each run.
3) Remove the edges with the highest misalignment coefficients, then go to Step 1 and run the dynamical model again, but this time using the new network without the removed edges.

Steps 1, 2, and 3, together, form what we call "cycle" or "round". In this paper, we study the influence of running the dynamical model multiple times per cycle. We also study the influence of removing different numbers of edges per cycle.

III. Theoretical results on flocking alignment

We also present analytical and argumentative study of the dynamics given by Equation (1).

A. Domain of the velocity vectors

A state at time $t + 1$ in which $\|\vec{v}_i(t)\| = 0$ for any i is a singularity. In order to deal with this problem, we need to restrict the parameter α. Once $\alpha > 0$ by definition, we show

that, if $\alpha < 0.5$ and $\|\vec{v}_i(0)\| = 1$ for all i, then $0 < \|\vec{v}_i(t)\| \leq 1$ for all i and $t > 0$. Thus, for any reasonable α and the initial conditions proposed in this paper, the velocity vectors are nonzero vector with norm less than or equal to 1.

Given the restrictions of the parameters and initial conditions, the following lemmas prove that $\|\vec{v}_i(t)\| \in (0, 1]$ for all i at any time t, guaranteeing the expected behavior of the model.

Lemma 1. *Given that $\|\vec{v}_i(0)\| \leq 1$, for all particle i, the norm of the velocity of every particle will not surpass* 1.

Proof. For $t = 0$, $\|\vec{v}_i(0)\| \leq 1$ from the statement.

Assume $\|\vec{v}_i(t)\| \leq 1$ for some t. Then,

$$\begin{aligned}
&\|\vec{v}_i(t+1)\| = \\
&\left\| \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \alpha \frac{1}{k_i} \sum_{j=1}^{N} \gamma_{ij} \left(\frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} - \frac{\vec{v}_i(j)}{\|\vec{v}_i(j)\|} \right) \right\| = \\
&\left\| (1-\alpha) \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \alpha \frac{1}{k_i} \sum_{j} \gamma_{ij} \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right\| \leq \\
&(1-\alpha) \left\| \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} \right\| + \alpha \frac{1}{k_i} \sum_{j} \gamma_{ij} \left\| \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right\| = \\
&(1-\alpha) + \alpha = 1 \implies \|\vec{v}_i(t+1)\| \leq 1 \quad (4)
\end{aligned}$$

Thus, by induction, $\|\vec{v}_i(t)\| \leq 1$ for all i, t. □

Lemma 2. *Given that $\|\vec{v}_i(0)\| > 0$ and $\alpha < 0.5$, for all particle i, the norm of the velocity of every particle is strictly greater than* 0 *for any time $t > 0$.*

Proof. For $t = 0$, $\|\vec{v}_i(0)\| > 0$ from the statement.

Assume that $\|\vec{v}_i(t)\| > 0$ for some t. We show by contradiction that $\|\vec{v}_i(t+1)\| > 0$.

If there exists $\alpha = \alpha_0$, $0 < \alpha_0 < 0.5$, such that $\|\vec{v}_i(t+1)\| = 0$, then

$$\begin{aligned}
&\vec{0} = (1-\alpha_0) \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \alpha_0 \frac{1}{k_i} \sum_{j} \gamma_{ij} \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \implies \\
&-\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} = \alpha_0 \left(-\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \frac{1}{k_i} \sum_{j} \gamma_{ij} \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right) \implies \\
&\left\| -\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} \right\| = \alpha_0 \left\| -\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} + \frac{1}{k_i} \sum_{j} \gamma_{ij} \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right\| \implies \\
&1 \leq \alpha_0 \left(\left\| -\frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} \right\| + \frac{1}{k_i} \sum_{j} \gamma_{ij} \left\| \frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} \right\| \right) = \\
&2\alpha_0 < 1 \implies 1 < 1. \quad (5)
\end{aligned}$$

By contradiction, such α_0 does not exist.

Thus, by induction, $\|\vec{v}_i(t)\| > 0$ for all i, t. □

B. Alignment of the velocity vectors

The core mechanism in our method is measuring small misalignments between connected particles and deciding which edge will be removed. A question that rises is whether the velocity vectors converge to a single point or not, that is, if the particles align or not. We show that perfect alignment of particles in the same connected component is most likely to happen. The system would also be in equilibrium if vectors are in perfect opposition to each other. Such condition would need not only very specific initial velocity vectors but also specific network configuration, thus this case is extremely unlikely.

To perform the particle-alignment study, we use a continuous approximation of the evolution equations,

$$\begin{aligned}
\frac{d}{dt}\vec{v}_i(t) = \frac{1 - \|\vec{v}_i(t)\|}{\|\vec{v}_i(t)\|}\vec{v}_i(t) + \\
\alpha \frac{1}{k_i} \sum_{j=1}^{N} \gamma_{ij} \left(\frac{\vec{v}_j(t)}{\|\vec{v}_j(t)\|} - \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} \right). \quad (6)
\end{aligned}$$

In the next equations, the summation indices i, j, and q always lie in the interval $[1, N]$. The intervals are dropped to simplify the notation.

We are not interested in the evolution of the unnormalized velocities but in their normalized forms. To improve readability, we set

$$\vec{x}_i(t) = \frac{\vec{v}_i(t)}{\|\vec{v}_i(t)\|} \text{ and} \quad (7)$$

$$z_i(t) = \|\vec{v}_i(t)\|. \quad (8)$$

Thus, the governing equations of the normalized velocities $\vec{x}_i(t)$ are

$$\begin{cases}
\frac{d}{dt}\vec{x}_i(t) &= \alpha \frac{1}{k_i z_i(t)} \sum_{q} \gamma_{iq} \Big(\vec{x}_q(t) - \big(\vec{x}_i(t) \cdot \vec{x}_q(t)\big)\vec{x}_i(t) \Big) \\
\frac{d}{dt} z_i(t) &= 1 - z_i(t) - \alpha \sum_{q} \gamma_{iq} \Big(1 - \vec{x}_i(t) \cdot \vec{x}_q(t) \Big)
\end{cases} \quad (9)$$

where $\cdot$ stands for the dot product operator.

Theorem 1. *The aligned state $\vec{x}_i(t) = \vec{x}_j(t)$, for all particle i that interacts with particle j, is stable in the sense of Lyapunov.*

Proof. We define the energy function E that reaches zero only when the aligned state is reached, and it increases as the velocities misalign. Let

$$\vec{u}_{ij} = \vec{x}_j - \vec{x}_i, \quad (10)$$

the candidate Lyapunov function is

$$E = \frac{1}{4} \sum_{i} \sum_{j} \gamma_{ij}\, \vec{u}_{ij} \cdot \vec{u}_{ij}. \quad (11)$$

Its derivative is

$$\frac{d}{dt}E = \sum_{i} \sum_{j} \frac{\partial}{\partial \vec{u}_{ij}} E \frac{d}{dt} \vec{u}_{ij}, \quad (12)$$

but

$$\frac{\partial}{\partial \vec{u}_{ij}} E = \frac{1}{4} \gamma_{ij} \left(2\vec{u}_{ij}\right) = \frac{1}{2} \gamma_{ij} \vec{u}_{ij}, \quad (13)$$

then

$$\frac{d}{dt}E = \frac{1}{2}\sum_i\sum_j \gamma_{ij}\vec{u}_{ij}\frac{d}{dt}\vec{u}_{ij} =$$

$$\frac{d}{dt}E = \frac{1}{2}\sum_i\sum_j \gamma_{ij}\big(\vec{x}_j - \vec{x}_i\big)\cdot\left(\frac{d}{dt}\vec{x}_j - \frac{d}{dt}\vec{x}_i\right). \quad (14)$$

However,

$$\vec{x}_i \cdot \frac{d}{dt}\vec{x}_i = \alpha\frac{1}{k_i z_i}\sum_q \gamma_{iq}\left(\vec{x}_i\cdot\vec{x}_q - \vec{x}_i\cdot\vec{x}_q\right) = 0, \quad (15)$$

then

$$\frac{d}{dt}E = -\sum_i\sum_j \gamma_{ij}\vec{x}_j\cdot\frac{d}{dt}\vec{x}_i =$$

$$-\sum_i\sum_j\sum_q \alpha\gamma_{ij}\gamma_{iq}\frac{1}{k_i z_i}\Big(\vec{x}_j\cdot\vec{x}_q - \big(\vec{x}_i\cdot\vec{x}_q\big)\big(\vec{x}_i\cdot\vec{x}_j\big)\Big) =$$

$$\sum_i\sum_j\sum_q \alpha\gamma_{ij}\gamma_{iq}\frac{1}{k_i z_i}\big(\vec{x}_i\cdot\vec{x}_q\big)\big(\vec{x}_i\cdot\vec{x}_j\big) -$$

$$\sum_i\sum_j\sum_q \alpha\gamma_{ij}\gamma_{iq}\frac{1}{k_i z_i}\vec{x}_j\cdot\vec{x}_q =$$

$$\alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\big(\vec{x}_i\cdot\vec{x}_j\big)\sum_q \gamma_{iq}\big(\vec{x}_i\cdot\vec{x}_q\big) -$$

$$\alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\sum_q \gamma_{iq}\big(\vec{x}_j\cdot\vec{x}_q\big) =$$

$$\alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\big(\vec{x}_i\cdot\vec{x}_j\big)\vec{x}_i\cdot\sum_q \gamma_{iq}\vec{x}_q -$$

$$\alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\vec{x}_j\cdot\sum_q \gamma_{iq}\vec{x}_q. \quad (16)$$

Let $\vec{x}_{\mathcal{N}_i} = \sum_q \gamma_{iq}\vec{x}_q$ be the summation over all the velocities of the neighbors of particle i, then

$$\frac{d}{dt}E = \alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\big(\vec{x}_i\cdot\vec{x}_j\big)\big(\vec{x}_i\cdot\vec{x}_{\mathcal{N}_i}\big) -$$

$$\alpha\sum_i\frac{1}{k_i z_i}\sum_j \gamma_{ij}\big(\vec{x}_j\cdot\vec{x}_{\mathcal{N}_i}\big) =$$

$$\alpha\sum_i\frac{1}{k_i z_i}\big(\vec{x}_i\cdot\vec{x}_{\mathcal{N}_i}\big)\vec{x}_i\cdot\sum_j \gamma_{ij}\vec{x}_j - \alpha\sum_i\frac{1}{k_i z_i}\vec{x}_{\mathcal{N}_i}\cdot\sum_j \gamma_{ij}\vec{x}_j =$$

$$\alpha\sum_i\frac{1}{k_i z_i}\big(\vec{x}_i\cdot\vec{x}_{\mathcal{N}_i}\big)\big(\vec{x}_i\cdot\vec{x}_{\mathcal{N}_i}\big) - \alpha\sum_i\frac{1}{k_i z_i}\big(\vec{x}_{\mathcal{N}_i}\cdot\vec{x}_{\mathcal{N}_i}\big). \quad (17)$$

But, given any vectors $\vec{a}, \vec{c} \in \mathcal{R}^D$, we have $\vec{a}\cdot\vec{c} \le \|\vec{a}\|\,\|\vec{c}\|$ and $\vec{a}\cdot\vec{a} = \|\vec{a}\|^2$,

$$\frac{d}{dt}E = \alpha\sum_i\frac{1}{k_i z_i}\big(\vec{x}_i\cdot\vec{x}_{\mathcal{N}_i}\big)^2 - \alpha\sum_i\frac{1}{k_i z_i}\big(\vec{x}_{\mathcal{N}_i}\cdot\vec{x}_{\mathcal{N}_i}\big) \le$$

$$\alpha\sum_i\frac{1}{k_i z_i}\big(\cancel{\|\vec{x}_i\|}\cancel{\|\vec{x}_{\mathcal{N}_i}\|}\big)^2 - \alpha\sum_i\frac{1}{k_i z_i}\|\vec{x}_{\mathcal{N}_i}\|^2 \le 0 \quad (18)$$

□

Remark 1. *Perfect alignment most likely happens since $\frac{d}{dt}E$ is zero only if x_i and $\vec{x}_{\mathcal{N}_i}$ are codirectional, that is, when all neighbors are either aligned or opposed to each other. While this condition does not hold, $\frac{d}{dt}E < 0$, and the aligned state is asymptotically stable.*

IV. Experimental results

In this section, we present an extensive set of experimental results conducted on different classes of computer-generated and real-world networks.

A. Illustrative example

In this subsection, we present illustrative examples of the application of our method. Input networks are built by employing the following methodology:

1) Given a desired average vertex degree $\langle k\rangle_{\rm des}$, each vertex i randomly chooses $\langle k\rangle_{\rm des}/2$ vertices j to connect to. For each j to be selected, j is taken from the same community of i with a probability $p_{\rm in}$, or accordingly taken from a different community with a probability $p_{\rm out} = 1 - p_{\rm in}$. The selection of the same vertex j twice or more is allowed, as it is also allowed for i to connect to itself. Also, j being selected by i does not prevent i being also selected by j.
2) After establishing all the connections, the network is simplified, in the sense that loops (self-connections) are removed and multiple edges that connect the same pair of vertices become a single, undirected edge. As a result, the actual average degree $\langle k\rangle$ may be reduced, but for large networks, it remains very close to the desired degree $\langle k\rangle_{\rm des}$.

For comparison purpose, in Figure 1, we also show the results of applying CFG and Louvain algorithms on a network with 4 communities, average degree $\langle k\rangle \approx 10$ and $p_{\rm in} = 0.66$. In this case, CFG achieves modularity $Q = 0.33$ and Louvain, $Q = 0.31$. We apply our method in this network, setting $\alpha = 0.1$. In each round, we run the dynamical system until $t = 100$ with 10 independent initial configurations. The most misaligned edge per round is removed until there is no more edges to remove. We choose the partition that yields the best modularity. The results of the iterative edge-removal process are better than those of CFG and Louvain, reaching $Q = 0.38$.

Figure 2 shows the modularity evolution over different rounds. Since communities are defined by connected components, the first edge-removal rounds just result in a single community, making the modularity score be very low. Those abrupt transitions reveal different components becoming completely disconnected from each other, i.e., they reveal those rounds in which the last edge that links two large components is removed. After achieving four major communities — and consequently the highest score —, further removal starts destroying them. Such a local optimum partition is the one that should be returned by the proposed iterative edge-removal method.

To further illustrate our method, we provide another simulation. We show (in Section III) that all the vertices in a connected component will align as time tends to infinity. As a result, one might wonder how our edge-removal strategy works in practice. To demonstrate its effectiveness, the method is

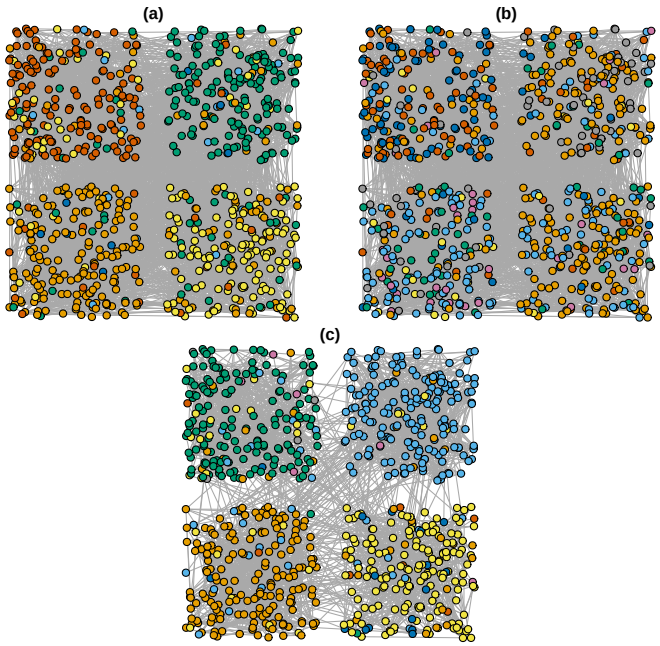


Fig. 1. Overview of the community-detection results. The input network has 4 communities, each one made of 200 vertices, $\langle k \rangle \approx 10$, and $p_{\text{in}} = 0.66$. The results of applying CFG (a) and Louvain (b) algorithms on the original network are shown. Communities are represented by different colors. In the last figure (c), we display the results after applying 1766 rounds of the proposed iterative edge-removal process. In this case, communities are just different connected components. In each cycle, 10 independent runs are performed and the most misaligned edge is removed. Only the remaining edges are shown in (c). We set $\alpha = 0.1$.

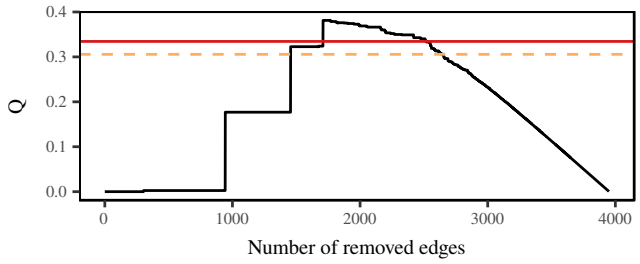


Fig. 2. Evolution of the modularity Q over the rounds of the proposed iteration edge-removal process. The input network and learning configuration are the same presented in Figure 1. Solid and dashed horizontal lines correspond to the modularity achieved by CFG and Louvain, respectively.

applied on a simpler community detection problem: detecting the 4 communities with the same size in a random clustered network that comprises 800 nodes with average degree $\langle k \rangle = 10$. Connections are randomly assigned such that every node has probability $p_{\text{in}} = 0.9$ of connecting to another node in its community. We run our method with 3-dimensional velocity vectors and $\alpha = 0.05$. For the sake of visualization, Figure 3 depicts the evolution of the normalized velocity vectors (projected in two dimensions) in different situations. Line colors represent communities, and the transparency decreases in function of time t. Plot (a) depicts 1000 iterations of our dynamical system in the original network. One can notice that all particles are aligned, but the velocity vectors of particles in different communities converge from different directions.

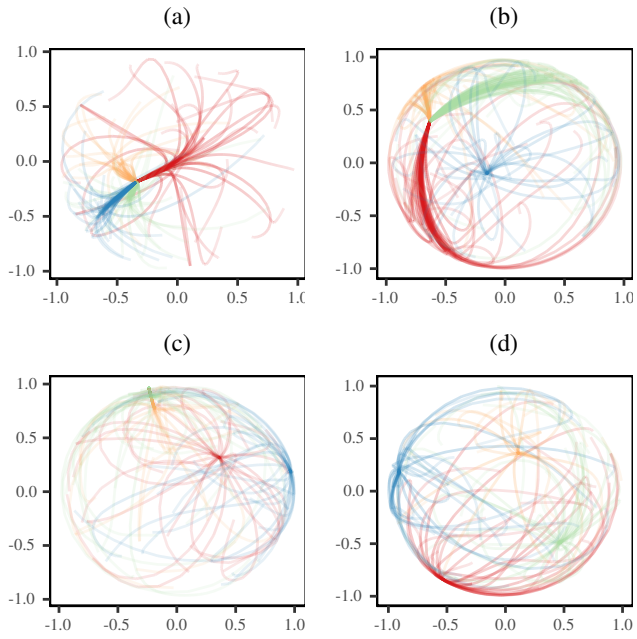


Fig. 3. Evolution of the dynamics governed by Equation 1The particle-alignment dynamical modelequation.2.1 in a network with 800 nodes, average degree $\langle k \rangle = 10$, 4 communities, and $p_{\text{in}} = 0.9$. For better visualization, even though the system has been simulated with $D = 3$ dimensions and 800 particles, only 200 random-selected normalized velocity vectors are shown projected in two dimensions. 1000 system iterations are run in each plot and the path of the velocity vectors are more opaque in the last iterations. Path colors match the four different communities. In plot (a), no edge was removed. In the remaining plots, 200 (b), 400 (c) and 600 (d) edges have been removed iteratively.

This phenomenon is utterly important, since it enables us to distinguish the communities. It also explains the difference between our method and Kuramoto-based ones [21], [22]. In the synchronization-based techniques, each element usually is a fixed low dimensional dynamical system. In the Kuramoto oscillator model, only a single real value is associated to each node, which corresponds to the phase. Thus, nodes can only synchronize "from two different directions". Using three or more dimensions in our method, we bring an infinitude of possible directions. In the same figure, we also show three snapshots of the edge-removal process. After running up to $t = 1000$, we remove the 10 edges with highest values of misalignment coefficient. We repeat this process until 600 edges have been removed. The evolution of the normalized velocity vectors at $t = 1, 2, \ldots, 1000$ after removing 200, 400, and 600 edges are illustrated in subplots (a), (b), and (c), respectively. After the removal of 200 edges, we observe that one of the communities disconnects from the others, becoming a single connected component, and thus, the velocity vectors converge to a different point. With 400 edges removed, another community detaches. And finally, after 600 removals, each community becomes a connected component of the network, achieving our goal.

B. Analysis of the evolution of misalignment coefficient

In the previous section, we claimed that the misalignment coefficients of edges connecting distinct communities become

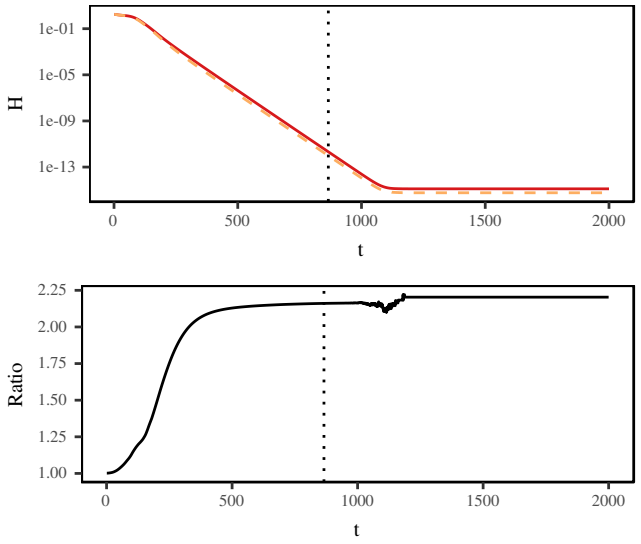


Fig. 4. Misalignment coefficient of the network presented in Figure 1: 4 communities, each one made of 200 vertices, $\langle k \rangle \approx 10$, and $p_{\text{in}} = 0.66$. The population of coefficients is obtained from 10 independent runs. Parameter $\alpha = 0.1$. Measurements are taken from a single round. Top plot: average misalignment coefficient evolution separated between intra- (dashed line) and inter-community (solid line) edges. Bottom plot: ratio between the misalignment coefficient of inter- and intra-community edges.

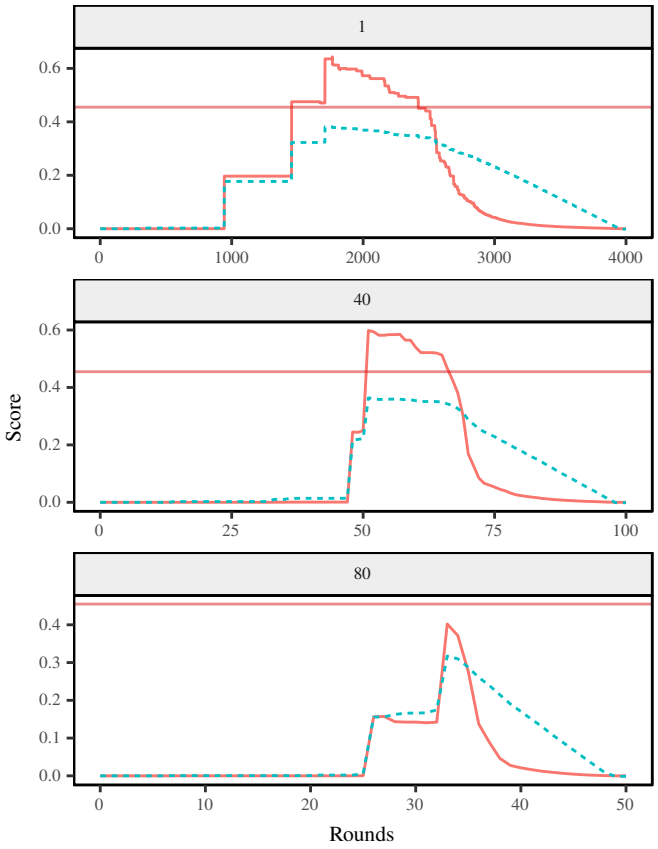


Fig. 5. ARI (solid lines) and modularity (dashed line) scores along different rounds. The input network is exactly the same one depicted in Figure 1: 4 communities, each one made of 200 vertices, $\langle k \rangle \approx 10$, and $p_{\text{in}} = 0.66$. Plots are arranged according to the amount of edges removed per round: 1, 20, and 80. For all runs, we set $\alpha = 0.1$ and the system runs until $t = 100$. The solid horizontal line indicates the ARI score obtained by the CFG method.

usually higher than those connecting vertices inside the same community. Let us now present, in Figure 4, how misalignment coefficients change over time.

In order to reduce the dependence on the initial condition – random assignment of velocities –, the population of misalignment coefficient values is obtained from 10 independent runs. The input network is the same for all of these runs. The network has 800 vertices and $\langle k \rangle \approx 10$, what gives a total of approximately 4000 edges.

In the top plot of Figure 4, we plot the average misalignment coefficient H grouped by intra- and inter-edges. As we can see, coefficients fall down quickly. Intra-edges, however, align faster than the inter-edges. In the bottom plot, we show the ratio between intra- and inter-edges. As expected, a greater proportion of intra-edges have lower misalignment coefficient. Moreover, after many iterations, the velocity vectors become almost identical for every pair of connected vertices. At this point, the finest machine representation of real numbers is reached. The vertical dotted line indicates the iteration t in which the average misalignment is lower than 10^{-12}. Any result beyond this point might be meaningless. Consequently, we should always stop the system earlier.

C. Analysis of the number of removed edges

Results of removing a different number of edges per round are presented in this section. An evaluation index is used in order to objectively quantify the accuracy of the set of obtained communities. Specifically, we selected the adjusted Rand index (ARI), which is the corrected-for-chance version of the Rand index [23]. It measures the similarity between the partition obtained from some algorithm and a reference partition. ARI generates values between -1 and 1. If two partitions match perfectly each other, it results in a value 1. On the other hand, it ensures a value close to 0 for a randomly-labelled partition or a partition that assigns all the elements into a single group, given that the reference partition has more than one group, of course. Negative values stand for some anti-correlation between the pair of partitions.

Real-world applications, however, do not provide such a reference partition, so we also employ the *modularity* score. Unlike ARI, the modularity score does not make any assumption on *a-priori* knowledge of the network, e.g., vertex labels. By placing the ARI score against the modularity score, we can check for any relationship between them, i.e., check for a relationship between an information available in a real application (modularity) and a robust measure based on reference partitions (ARI).

Firstly, let us consider the same network presented in Figure 1, with $p_{\text{in}} = 0.66$. ARI and modularity scores along different rounds are presented in Figure 5, arranged according to the number of edges removed per round.

Even when 40 edges ($\approx 1\%$ of the total number of edges) are removed per round, our method achieves higher ARI score than the CFG method: 0.64 (1 edge) and 0.60 (40 edges) against 0.45. Removing fewer edges per round is

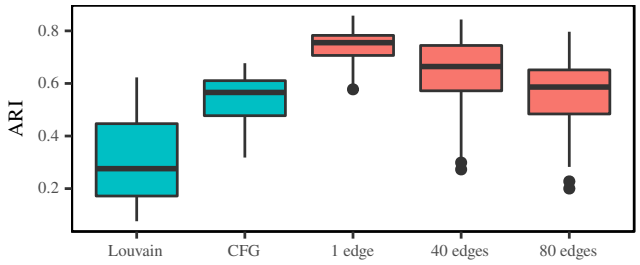


Fig. 6. Results of removing 1, 40 and 80 edges per round on a population of 50 different networks. Each network has 4 communities, each one made of 200 vertices, $\langle k \rangle \approx 10$, and $p_{\text{in}} = 0.66$. Boxes in blue are the results of applying CFG and Louvain on the network. The remaining boxes (in red) are obtained by applying the proposed technique removing different amounts of edges. The partition that yields the best modularity is chosen. 10 runs per round are performed, and parameter $\alpha = 0.1$ for all simulations. We stop the system at $t = 100$.

slightly better. Removing a larger fraction of the edges is less expensive though, for it demands fewer rounds to complete. However, we experience a significantly drop in the score if the fraction of removed edges per round is too big (around 2%.)

Moreover, a desirable relationship is noticeable here: the highest modularity matches relatively high ARI scores. This result is useful for establishing when we should stop the edge-removal process and where the optimal round is, i.e., the round after which we are likely to find a good partition: stopping the process just before the modularity starts decreasing and taking such highest-modularity's network, using the connected components as the final set of communities. Although the results of Figure 5 come from just one network instance, the same overall pattern is still present in other instances, even using different network parameters, as we are going to see in the remaining results presented in this paper.

In Figure 6, we present a comparison between removing 1, 40, and 80 edges per round on a population of 50 different networks. In fact, removing a smaller fraction of the edges yields better results. In addition, the variability of results is reduced when removing less edges per round. However, the proposed edge-removal process leads to considerably better results compared to the application of CFG or Louvain on the original network.

D. Computational complexity

The edge-removal approach presented here is somewhat similar to the edge-betweenness-based algorithm proposed by Newman and Girvan [15]. The misalignment coefficient of an edge measures how different the incident vertices are in terms of their velocity vector. Edge betweenness, in turn, measures the relevance of an edge in terms of the number of shortest paths that include such an edge. Both concepts ultimately try to identify edges that connect distinct communities. An important drawback of the edge-betweenness approach is, however, its cubic time complexity. Precisely, for a network of m edges and n vertices, the time complexity is $O(m^2 n)$, or $O(n^3)$ for sparse networks, in which case $m \sim n$ [9], [12], [15], [17]. Therefore, it is necessary to have a discussion about the complexity of the approach proposed here.

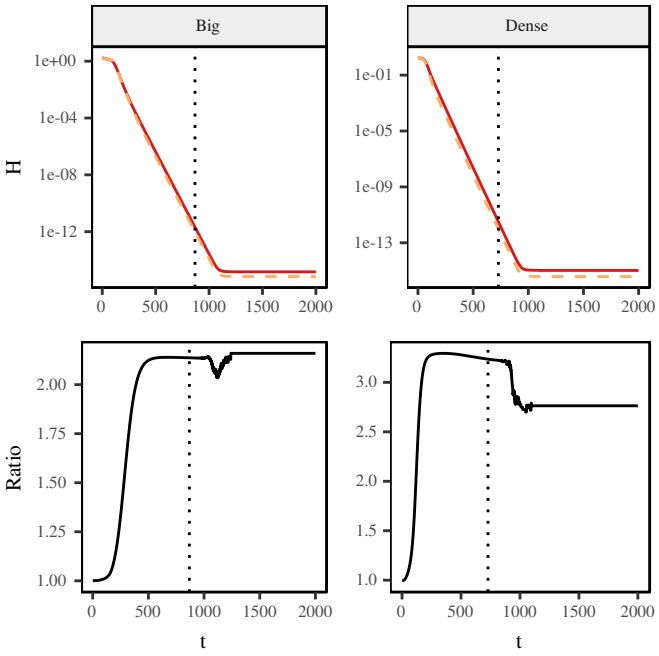


Fig. 7. Misalignment coefficient of a big and a dense network: both of them with 4 communities and $p_{\text{in}} = 0.66$. The big network has 2000 vertices in each community, while the dense has 200. The average degree in the big network is $\langle k \rangle \approx 10$, while in the dense one, $\langle k \rangle \approx 20$. The population of coefficients is obtained from 10 independent runs. Parameter $\alpha = 0.1$. Measurements are taken from a single round. Top plots: average misalignment coefficient evolution separated between intra- (dashed line) and inter-community (solid line) edges. Bottom plots: ratio between the misalignment coefficient of inter- and intra-community edges.

Four variables are relevant to our time-complexity analysis: the number n of vertices; the number m of edges; the total amount of time-steps t_{total} per run; and the number of edge-removal rounds n_{rounds} required to obtain an appropriate partition of the network. The dimensionality of the velocity vectors is constant, three dimensions are employed in our experiments, so it is not relevant to this analysis. Our hypothesis is that both t_{total} and n_{rounds} can be invariant no matter how big the network is.

In the case of removing just a single edge per round, n_{rounds} gets the order of m. However, the experiments presented in the previous section indicate that it is still possible to obtain satisfactory results by removing a fraction of all the edges per round, in which case n_{rounds} becomes constant. For instance, removing approximately 1% of the edges per round requires around 50 rounds to achieve good results. Also, concerning the number of runs per round, if one decides to perform a set of runs per round in order to reduce the influence of the initial random configuration, the number is still relatively small and does not scale with neither n nor m.

In order to provide evidence that supports our hypothesis – neither t_{total} nor n_{rounds} scales with the network size – , we present some results on bigger and denser networks. Particularly, a network that has 10 times more vertices than those studied in the previous section and a network that has twice more edges while keeping the same number of vertices.

In Figure 7, we present the evolution of the coefficients

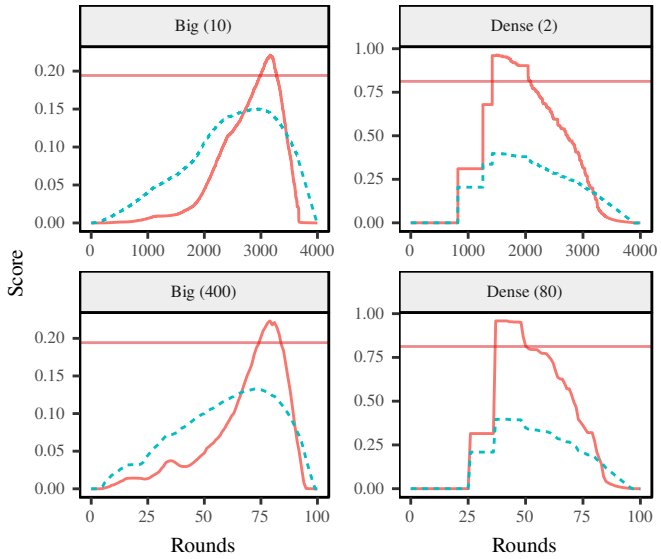


Fig. 8. ARI (solid lines) and modularity (dashed line) scores along different rounds. Input networks are exactly the same ones depicted in Figure 7. Plots are arranged according to the amount of edges removed per round: $\approx 0.025\%$ and 1%. The exact number of edges removed per round is indicated between parentheses. For all runs, we set $\alpha = 0.1$ and the system runs until $t = 80$ for the dense network and $t = 250$ for the big network. The solid horizontal line indicates the ARI score obtained by the CFG method.

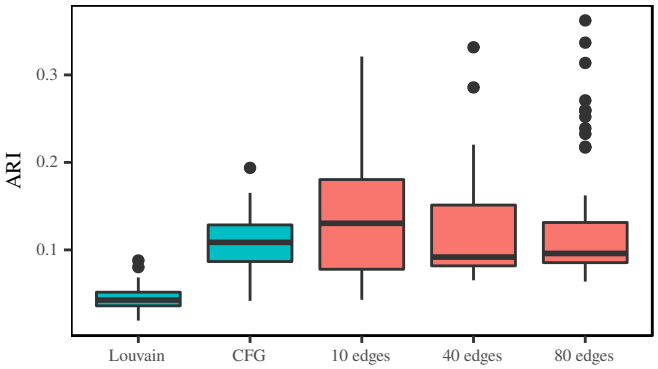


Fig. 9. Results of removing 10, 40 and 80 edges per round on a population of 50 different networks. Each network has 4 communities, containing 800, 400, 200, and 100 vertices, $\langle k \rangle \approx 10$, and $p_{\text{in}} = 0.66$. Boxes in blue are the results of applying CFG and Louvain on the network. The remaining boxes (in red) are obtained by applying the proposed technique removing different amounts of edges. The partition that yields the best modularity is chosen. 10 runs per round are performed, and parameter $\alpha = 0.1$ for all simulations. We stop the system at $t = 250$.

in the first round. We notice that increasing the network size or the density has small effects on the amount of time-steps needed to obtain satisfactory separation. Interestingly, the bigger network requires a little more iterations while the denser one, a little less. Therefore we can estimate that, for an arbitrarily large network, t_{total} remains almost the same, or at least do not scale linearly with the network size.

Furthermore, we analyse the scores along different rounds by running up to the iteration $t = 80$ for the dense network and $t = 250$ for the big network. Results are plotted in Figure 8. Both the misalignment coefficient evolution and the ARI score follow similar shapes compared with the results of smaller networks presented in the previous sections.

Since the summation in Equation 1The particle-alignment dynamical modelequation.2.1 takes place on the set of edges, the time complexity for a single dynamical iteration t is $O(m)$. Seeing that t_{total} does not scale with the network size, the entire run still takes $O(m)$. In addition, at the end of each round (a single run or a small set of runs), we need to target the highest-misaligned edges, what requires sorting the set of edges and usually takes $O(m \log m)$. Since n_{rounds} does not scale with the network size either, then all the edge-removal process has a quasilinear time complexity: $O(m \log m)$ if $m \gg n$, or just $O(n \log n)$ for sparse networks.

However, it is important to emphasize that the cost $O(m \log m)$ holds only from the asymptotic point of view, when networks become very large. For small- or medium-sized networks, the cost caused by t_{total} and n_{rounds} may become noticeable.

E. Experiments on unbalanced-communities networks

All the experiments presented so far are performed on a class of balanced networks, whose communities have the same size. In order to provide more-realistic examples, let us introduce now some results on networks whose communities have different number of vertices. These results are shown in Figure 9.

We notice now a considerable drop in the quality of the partitions. Still, compared to traditional algorithms like CFG or Louvain applied to the original network, the proposed edge-removal process performs well. The performance of the proposed method, however, varies greatly. One possible explanation is that the fixed number of iterations $t = 250$ is not ideal. The study of heuristics for the system's stop condition are left for future works.

F. Experiments on Lancichinetti benchmark

Since real-world networks usually have heterogeneous degree distribution, we also apply our technique on the benchmark of Lancichinetti *et al.* [24]. Such benchmark produces networks in which both degree and community size distributions follow power law functions with arbitrary exponents. Motivated by typical values found in natural systems, we choose exponent 2 for the degree distribution and 1 for the size of communities. Community sizes are in $[3, 80]$. The generated networks have a mixing parameter μ that controls the fraction of links between nodes of different communities. To assess our method, we use networks with 1000 nodes, average degree 10, and maximum degree 40. The mixing parameter μ varies between 0.1 and 0.6.

We compare our method against CFG and Louvain in 30 independent trials. The performance is measured in terms of the normalized mutual information index, which measures the similarly of the predicted partition against the expected one. Moreover, such index is suggested by the authors of the benchmark. CFG and Louvain have no parameter, while in our technique we set $\alpha = 0.05$. For stopping criterion, we run

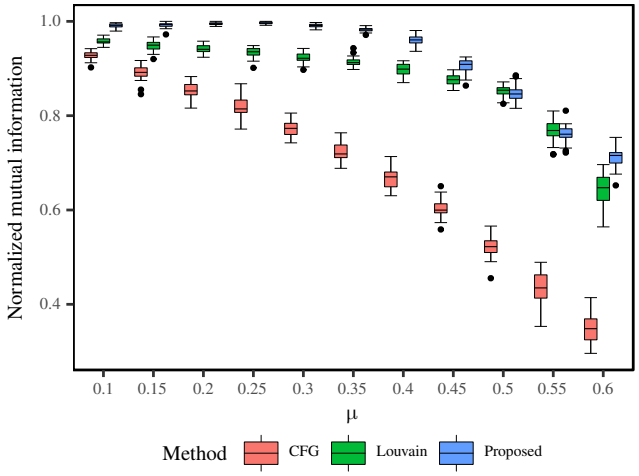


Fig. 10. Comparison of the normalized mutual information index obtained by three community detection algorithms—our proposed method, CFG and Louvain—in the Lancichinetti *et al.* benchmark [24]. Results are taken from 30 independent trials for each value of the mixing parameter μ. Networks contain 1000 nodes, average degree 10 (maximum degree 40). Degree and community size distribution follow power law functions with exponents 2 and 1, respectively. (Community-size range is $[3, 80]$.)

TABLE I
MODULARITY SCORE OF COMMUNITY DETECTION METHODS IN 4 CLASSIC REAL-WORLD DATASETS. OPTIMAL MODULARITY IS SHOWN FOR COMPARISON.

Method	Dolphin	Football	Karate	Political Books
Proposed	0.529	0.605	0.419	0.527
Amiri [28]	0.515	0.597	0.417	0.518
Honghao [29]	0.529	0.605	0.420	0.527
Song [30]	-	0.531	0.362	0.463
Optimal	0.529	0.605	0.420	-

the system until the maximum change in each projection $\hat{v}_{i,d}$ is less than 10^{-3} and remove the edge with higher misalignment coefficient. We interpret each connected component as a community. We repeat the edge-removal process until the modularity of the partitioning starts decreasing.

Figure 10 reveals that our method performs significantly better than CFG and Louvain. Beyond the mark $\mu = 0.5$, communities are no longer defined in a strong sense since each node has more neighbors in other communities than in its own.

G. Experiments on real-world datasets

We apply our algorithm to four well-known real-world networks: Zachary's karate club [25], the bottlenose dolphins social network [26], the American College Football [27], and the Krebs' books on US politics[1].

Table I presents the best modularity scores obtained by our method. We set $\alpha = 0.1$, 30 independent runs, and vary the fraction of removed edges in 1%, 2%, $\ldots$, 10%. Also, we vary the number of iterations $t \in [30, 70]$. We compare our results against three other bio-inspired optimization methods. (Missing results have not been measured by the authors of the original paper.) Also, the optimal value of modularity is calculated using an exhaustive search in all possible partitions. Modularity optimization is an NP-complete problem, and known algorithms have exponential time complexity [15]. (We were not able to find out the optimal partition of the *political books* network.)

We observe that our method reached either optimal or nearly-optimal modularity scores. Although the ant-colony-based technique [29] had similar performance, our algorithm has lower computational cost because it explores the solution space in a greedy manner.

[1]This network is unpublished and can be found at `http://www-personal.umich.edu/~mejn/netdata/`

V. CONCLUSIONS

Throughout an extensive set of experiments presented in this paper, we see that the proposed flocking-like dynamical system and the iterative edge-removal process performs well in many scenarios. The decentralized, self-organizing dynamical model is robust, thus applicable on a wide variety of networks.

The concept of misalignment coefficient defined here is, in some sense, similar to the concept of edge betweenness. High-misaligned edges are supposed to link distinct communities. However, the cost $O(n^3)$ (on sparse networks) of the method proposed by Newman and Girvan [15] can be prohibitive for its application in large networks. On the other hand, we claim that our edge-removal process is asymptotically quasilinear: $O(n \log n)$, which is quite attractive for large-network community detection.

In further works, we will study heuristics to find out good values of the number of iterations and removed edges per round.

ACKNOWLEDGMENTS

This research work was supported by the State of São Paulo Research Foundation (FAPESP) (Projects 13/08666-8, 13/25876-6, and 15/50122-0), by the Brazilian National Research Council (CNPq) (Project 303012/2015-3), and by the German Research Foundation (DFG) (Project IRTG/GRK 1740).

R.A. Gueleri and F.A.N. Verri programmed and performed the computer experiments, as well as produced the resultant plots and figures. F.A.N. Verri performed the analytical study. All authors contributed on conceiving the method, planning the experiments, and writing this manuscript. L. Zhao, in addition, supervised all this research work.

REFERENCES

[1] F. A. N. Verri, P. R. Urio, and L. Zhao, "Network unfolding map by vertex-edge dynamics modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, 2016.

[2] C. Pizzuti, "Evolutionary Computation for Community Detection in Networks: a Review," *IEEE Transactions on Evolutionary Computation*, vol. X, no. X, pp. 1–1, 2017.

[3] Q. Cai, M. Gong, S. Ruan, Q. Miao, and H. Du, "Network structural balance based on evolutionary multiobjective optimization: A two-step approach," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 903–916, 2015.

[4] P. Lv, J. Zhang, and H. Zhang, "A social network graphics segmentation algorithm based on community-detection," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 619–623.

[5] Z.-K. Gao, Y.-X. Yang, P.-C. Fang, N.-D. Jin, C.-Y. Xia, and L.-D. Hu, "Multi-frequency complex network from time series for uncovering oil-water flow structure," *Scientific Reports*, vol. 5, 2015.
[6] C. Thiemann, F. Theis, D. Grady, R. Brune, and D. Brockmann, "The structure of borders in a small world," *PLoS ONE*, vol. 5, no. 11, 2010.
[7] C. Zhong, S. M. Arisona, X. Huang, M. Batty, and G. Schmitt, "Detecting the dynamics of urban structure through spatial network analysis," *International Journal of Geographical Information Science*, vol. 28, no. 11, pp. 2178–2199, 2014.
[8] E. M. Heemskerk and F. W. Takes, "The corporate elite community structure of global capitalism," *New Political Economy*, vol. 21, no. 1, pp. 90–118, 2016.
[9] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, 2004.
[10] M. G. Quiles, E. E. N. Macau, and N. Rubido, "Dynamical detection of network communities," *Scientific Reports*, vol. 6, 2016.
[11] T. C. Silva and L. Zhao, *Machine learning in complex networks*. Springer International Publishing, 2016.
[12] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.
[13] M. Mitrović and B. Tadić, "Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities," *Physical Review E*, vol. 80, no. 2, 2009.
[14] E. Agliari and F. Tavani, "The exact Laplacian spectrum for the Dyson hierarchical network," *Scientific Reports*, vol. 7, p. 39962, 2017.
[15] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, 2004.
[16] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Physical Review E*, vol. 84, no. 6, 2011.
[17] ——, "Community detection algorithms: a comparative analysis," *Physical Review E*, vol. 80, no. 5, 2009.
[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, 2008.
[19] K. H. Nagai, Y. Sumino, R. Montagne, I. S. Aranson, and H. Chaté, "Collective motion of self-propelled particles with memory," *Physical Review Letters*, vol. 114, no. 16, 2015.
[20] T. Vicsek and A. Zafeiris, "Collective motion," *Physics Reports*, vol. 517, pp. 71–140, 2012.
[21] E. Oh, K. Rho, H. Hong, and B. Kahng, "Modular synchronization in complex networks," *Physical Review E*, vol. 72, no. 4, p. 047101, 2005.
[22] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, "Synchronization in complex networks," *Physics Reports*, vol. 469, no. 3, pp. 93–153, 2008.
[23] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.
[24] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, p. 046110, 2008.
[25] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.
[26] O. B. P. H. E. S. D. Lusseau, K. Schneider and S. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.
[27] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[28] B. Amiri, L. Hossain, J. W. Crawford, and R. T. Wigand, "Community detection in complex networks: Multi–objective enhanced firefly algorithm," *Knowledge-Based Systems*, vol. 46, no. Supplement C, pp. 1 – 11, 2013.
[29] C. Honghao, F. Zuren, and R. Zhigang, "Community detection using ant colony optimization," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 3072–3078.
[30] A. Song, M. Li, X. Ding, W. Cao, and K. Pu, *International Journal of Computer Science*, no. 1, pp. 37–44.

CHAPTER

7

CONCLUSION

Learning methods based on complex networks and collective dynamics encompass three equally important steps: network formation, information retrieval, and decision. In the first step, a graph is constructed from the input data. Then, the simulation of a distributed collective dynamics generates rich information about the data. Finally, the collected information is used to solve the underlying machine-learning problem, whether it is a classification or a clustering task. In this research, I have explored collective dynamics in machine learning tasks, both in unsupervised and semi-supervised scenarios. My contributions cover a proposal of a network construction method, proposals of new collective systems, the mathematical understanding of the information generated by these dynamical systems, and applications in many machine learning problems.

7.1 Concluding remarks

In the following subsections, each topic of the research is commented.

7.1.1 Particle competition and edge-centric dynamics

In Chapter 2, it is presented the paper that introduces a transductive SSL technique based on an edge-centric dynamical system on complex networks. First, the input data should be transformed using any network formation technique that produces an undirected, unweighted, simple graph. Then, the proposed Labeled Component Unfolding (LCU) system runs on this network. Finally, we employ the information from the system to classify unlabeled data.

In summary, the LCU system consists of particles that compete for edges in the network. When a particle passes through an edge, it increases its class dominance over the edge while decreasing other classes' dominance. Three biologically inspired dynamics – walking, absorption and production – provide a scenario of competition and cooperation. Labels are assigned

according to the dominant class over the edges, inherently unfolding the original network by grouping edges dominated by the same class.

Rigorous studies have been performed on the LCU system to justify its deterministic implementation. Such implementation brings advantages over its stochastic counterpart. The time complexity of the deterministic one does not depend on the number of particles, so we allow a stable transductive SSL technique with a subquadratic order of complexity.

Computer simulations show that the proposed technique achieves a good classification accuracy and it is suitable for situations where a small number of labeled samples are available. Another interesting feature of the proposed model is that it directly provides the overlapping information of each vertex or a subset of vertices.

In Chapter 3, the paper with studies on the advantages of an edge-centric collective dynamics in semi-supervised learning tasks is exposed. A slightly different version of the LCU, called Edge Domination System (EDS) is a particle competition system that can be used to solve data classification tasks. We derived a similar but vertex-based version of the system, Vertex Domination System (VDS), and studied their differences.

The systems EDS and VDS can be simplified without impairing its performance in learning problems. Both models offer similar performance in classification and have the same time complexity order. EDS, however, not only acquires more information about the overlapping area with mixed rival vertices but also has better exploration behavior. Such results confirm the behavior of the LCU system. The edge-centric approach has advantages regarding the running time as well. It has lower average convergence time than the vertex-centric approach. Therefore, the evidence points towards favoring EDS over VDS.

7.1.2 Feature–sample networks

In Chapter 4, we present a transductive classification method for solving positive-unlabeled (PU) learning problems, that is, where only a few positive-labeled and many unlabeled samples are given as input. The learning system is based on a collective dynamic of moving particles that runs over a bipartite network. The bipartite network, called feature–sample network, is built using a novel network formation method that does not rely on the similarity of input samples.

The first step of the algorithm maps the input data into a sparse binary representation and, afterward, into a bipartite complex network whose vertices represent samples and attributes. The existence of an edge depends on the presence of a certain feature in a certain sample. Then, a Markov Chain that outputs positive-class pertinence levels for the unlabeled samples is produced. Finally, we classify the unlabeled samples based on the pertinence levels and the positive-class prior probability.

The proposed scheme offers high performance on PU learning problems, even with few

labeled samples. The learning model itself present some major limitations, but the network formation technique is simple and opens room for the development of new collective dynamics in bipartite networks for machine learning.

Chapter 5 includes a paper that proposes an unsupervised feature learning mechanism for the feature–sample networks. A multi-objective optimization process selects a set of new vertices that correspond to new features to produce an enhanced version of the network. The enhanced network contains more information that is exploited to improve the performance of machine learning methods.

7.1.3 Flocking-like system for clustering

Chapter 6 consists of the paper that proposes a flocking-like dynamical system together with an iterative edge-removal process to solve community detection problems. In the model, each vertex is an aligning particle and carries a state. They start pointing to a random direction and, as the process evolves, progressively turn themselves toward the same direction of their neighbors. After some iterations, the edge (or group of edges) that connects the least aligned pair of vertices is supposed to link distinct communities and is removed from the network. Inter-community edges are expected to be removed, thus the network becomes partitioned into disconnected components. These components are the discovered communities of the network.

An extensive set of experiments shows that the method performs well in many scenarios. The decentralized, self-organizing dynamical model is robust, thus applicable to a wide variety of networks. Moreover, mathematical analysis proves the desirable behavior of the system, including the domain of the states and the likely alignment of the particles.

7.2 Scientific contributions

During the doctorate period, the following papers have been published, accepted or submitted as a result of this work:

- Published:

  1. Filipe Alves Neto, Liang Zhao, “On the data classification using complex network entropy”, BDBComp Proceedings (Encontro Nacional de Inteligência Artificial e Computacional), v. 1, p. 53-58, São Carlos, Brazil, 2014.

  2. G. Furquim, F. A. Neto, G. Pessin, J. Ueyama, J. P. de Albuquerque, M. Clara, E. M. Mendiondo, V. C. B. de Souza, P. de Souza, D. Dimitrova, T. Braun, “Combining Wireless Sensor Networks and Machine Learning for Flash Flood Nowcasting”, Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops, v. 1, p. 67-72, Victoria, BC, Canada, 2014.

3. Paulo Roberto Urio, Filipe Alves Neto Verri, Liang Zhao, "Semi-supervised Learning by Edge Domination in Complex Networks", Proceedings of the 11th International Conference on Natural Computation, v. 1, p. 516-521, Zhangjiajie, China, 2015.
4. Paulo Roberto Urio, Filipe Alves Neto Verri, Liang Zhao, "Semi-supervised Classification by Particle Competition in Complex Network's Edges", International Journal of Pattern Recognition and Artificial Intelligence, v. 30, p. 1660006, 2016.
5. Filipe Alves Neto Verri, Liang Zhao, "Random Walk in Feature–Sample Networks for Semi-supervised Classification", Proceeding of the 5th Brazilian Conference on Intelligent Systems, v. 1, p. 235-240, Recife, Brazil, 2016.
6. Paulo Roberto Urio, Filipe Alves Neto Verri, Liang Zhao, "Features of edge-centric collective dynamics in machine learning tasks", Proceedings of the 6th International Conference on Nonlinear Science and Complexity, online <https://ssl4799.websiteseguro.com/swge5/PROCEEDINGS/>, São José dos Campos, SP, 2016.
7. Filipe Alves Neto Verri, Paulo Roberto Urio, Liang Zhao, "Network Unfolding Map by Edge Dynamics Modeling", IEEE Transactions on Neural Networks and Learning Systems, v. 29, no. 2, p. 405-418, 2018.

- Accept for publication:

1. Filipe Alves Neto Verri, Paulo Roberto Urio, Liang Zhao, "Advantages of Edge-Centric Collective Dynamics in Machine Learning Tasks", Journal of Applied Nonlinear Dynamics, v. 7, no. 3, September, 2018 (special issue, not published yet).

- Submitted:

1. Filipe Alves Neto Verri, Roberto Alves Gueleri, Qiusheng Zheng, Junbao Zhang, Liang Zhao, "Network community detection via iterative edge removal in flocking-like system", Submitted to IEEE Transactions on Evolutionary Computation, 2018. Preprint, `arXiv:1802.04186 [cs.SI]`, <https://arxiv.org/abs/1802.04186>.
2. Filipe Alves Neto Verri, Renato Tinós, Liang Zhao, "Feature learning in feature-sample networks using multi-objective optimization". Submitted to the IEEE Congress on Evolutionary Computation, 2018. Preprint, `arXiv:1710.09300 [cs.AI]`, <https://arxiv.org/abs/1710.09300>.

The resulting publications have rendered many contributions to the areas of machine learning and complex systems. Regarding my individual ideas and achievements, I highlight the following contributions:

1. Proposal and analytical study of a new collective system of competing particles, the Labeled Component Unfolding (LCU) system (Chapter 2). The LCU system shifts the

traditional vertex-centric dynamics to a more informative edge-centric dynamics. Moreover, it is the first particle competition system applied in machine learning task that has deterministic behavior.

2. Idealization and part of the conduction of the analytical and experimental studies on the differences between edge- and vertex-centric particle competition (Chapter 3). Results show several advantages of the edge-centric model, including the ability to acquire more information about overlapping areas, a better exploration behavior, and a faster convergence time.

3. Proposal of a new network formation technique that is not based on similarity, called feature–sample networks (Chapter 4). In datasets with binary features, the method is lossless. In other words, in that case, the method does not lose information of the input data. The formation method has low computation cost of construction and update.

4. Analytical investigation of the flocking-like system of aligning particles (Chapter 6). The understanding of the mathematical properties of the system is needed to guarantee the expected behavior in community detection tasks.

7.3 Future works

I list the major topics that could be investigated in future works:

1. The relationship between particle competition models and reinforced random walks (PEMANTLE, 2007). Reinforced random walks are a challenging topic in statistics, but some theoretical results could give insights about the application of particle competition in machine learning.

2. The relationship between edge-centric particle competition and edge-centric community detection (AHN; BAGROW; LEHMANN, 2010; LIU *et al.*, 2016). The proposed LCU system can benefit from the study of link communities and edge label propagation.

3. Mathematical and experimental study of the convergence of LCU and VDS. Although maximum competition systems ($\lambda = 1$) have fixed points in networks with binary labels, the convergence is not guaranteed analytically. Stability analysis can be carried out using either Lyapunov direct method or the Jacobian matrix of the system. Results may be further extended to multi-label problems and for lower levels of competition ($0 < \lambda < 1$).

4. Proposal of adaptations of edge-centric systems, LCU and VDS, to work on directed and/or weighted networks. The change of the characteristics of the input graph could lead to unexpected differences in the dynamics.

5. Explicit exploitation of the information retrieved by LCU and VDS. Edge-centric systems can retrieve more information than their vertex-centric counterpart, especially about overlapping areas. I speculate that a better designed classification mechanism will further increase accuracy of those methods.

6. Proposal of collective systems that explicit retrieve information from bipartite networks. Mutualistic systems (LEVER *et al.*, 2014; JIANG *et al.*, 2018) exhibit interesting dynamics and have several properties that can be used for machine learning.

7. Application of feature–sample networks in real-time problems. Since addition and removal of samples in the network is cheap, methods can be developed to deal with concept drift, outlier detection, data stream classification, and active learning.

8. Mathematical study of alignment of communities in the flocking-like system. It is shown that all particles align eventually, but no analytical study has been carried about the partial and hierarchical alignment of the communities. Such study would provide better heuristics for the stopping conditions of the system itself and the removal of edges.

9. Application of the flocking-like system in semi-supervised learning problems. The flocking-like system can be adapted to benefit from label information, deriving new learning models.

BIBLIOGRAPHY

AGRIOMALLOS, I.; DOLTSINIS, S.; MITSIONI, I.; DOULGERI, Z. Slippage detection generalizing to grasping of unknown objects using machine learning with novel features. **IEEE Robotics and Automation Letters**, v. 3, n. 2, p. 942–948, 2018. Citation on page 15.

AHN, Y. Y.; BAGROW, J. P.; LEHMANN, S. Link communities reveal multiscale complexity in networks. **Nature**, Nature Publishing Group, v. 466, n. 7307, p. 761–764, 2010. Citation on page 89.

ALDAHDOUH, A. A.; OSÓRIO, A. J.; CAIRES, S. Understanding knowledge network, learning and Connectivism. **International Journal of Instructional Technology and Distance Learning**, v. 12, n. 10, p. 3, 2015. Citation on page 15.

ARENAS, A.; DÍAZ-GUILERA, A.; PÉREZ-VICENTE, C. J. Synchronization reveals topological scales in complex networks. **Phys. Rev. Lett.**, American Physical Society, v. 96, p. 114102, 2006. Citation on page 18.

BÄCK, T. **Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms.** Oxford, UK: Oxford University Press, 1996. 314 p. Citation on page 17.

BANERJI, M.; LAHAV, O.; LINTOTT, C. J.; ABDALLA, F. B.; SCHAWINSKI, K.; BAMFORD, S. P.; ANDREESCU, D.; MURRAY, P.; RADDICK, M. J.; SLOSAR, A.; SZALAY, A.; THOMAS, D.; VANDENBERG, J. Galaxy zoo: reproducing galaxy morphologies via machine learning. **Monthly Notices of the Royal Astronomical Society**, v. 406, n. 1, p. 342–353, 2010. Citation on page 15.

BELKIN, M.; NIYOGI, P.; SINDHWANI, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. **J. Mach. Learn. Res.**, v. 7, p. 2399–2434, 2006. Citation on page 17.

BERTINI, J. R.; ZHAO, L.; MOTTA, R.; LOPES, A. de A. A nonparametric classification method based on k-associated graphs. **Information Sciences**, v. 181, n. 24, p. 5435 – 5456, 2011. Citation on page 19.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. Citation on page 15.

BREVE, F.; ZHAO, L. Fuzzy community structure detection by particle competition and cooperation. **Soft Computing**, v. 17, n. 4, p. 659–673, 2013. Citation on page 18.

CHENG, L.; PAN, S. J. Semi-supervised Domain Adaptation on Manifolds. **IEEE Transactions on Neural Networks and Learning Systems**, v. 25, n. 12, p. 2240–2249, 2014. Citation on page 17.

CUPERTINO, T. H.; GUELERI, R.; ZHAO, L. A semi-supervised classification technique based on interacting forces. **Neurocomputing**, v. 127, p. 43 – 51, 2014. Advances in Intelligent Systems. Citation on page 18.

CUPERTINO, T. H.; HUERTAS, J.; ZHAO, L. Data clustering using controlled consensus in complex networks. **Neurocomputing**, v. 118, p. 132 – 140, 2013. Citation on page 18.

DAMIANCE, A. P.; ZHAO, L.; CARVALHO, A. C. A dynamical model with adaptive pixel moving for microarray images segmentation. **Real-Time Imaging**, v. 10, n. 4, p. 189 – 195, 2004. Imaging in Bioinformatics: Part III. Citation on page 18.

DEVINE, T. R.; GOSEVA-POPSTOJANOVA, K.; MCLAUGHLIN, M. Detection of dispersed radio pulses: a machine learning approach to candidate identification and classification. **Monthly Notices of the Royal Astronomical Society**, v. 459, n. 2, p. 1519–1532, 2016. Citation on page 15.

DORIGO, M.; BIRATTARI, M. Ant colony optimization. In: ____. **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 36–39. Citation on page 17.

FORTUNATO, S. Community detection in graphs. **Physics Reports**, Elsevier B.V., v. 486, n. 3-5, p. 75–174, 2010. Citation on page 18.

GROSSBERG, S. Competitive learning: From interactive activation to adaptive resonance. **Cognitive Science**, v. 11, n. 1, p. 23 – 63, 1987. Citation on page 19.

GUELERI, R. A.; CUPERTINO, T. H.; CARVALHO, A. C. P. L. F. de; ZHAO, L. A flocking-like technique to perform semi-supervised learning. In: **2014 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2014. p. 1579–1586. Citation on page 21.

HAYES, A. T.; DORMIANI-TABATABAEI, P. Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 2002. v. 4, p. 3900–3905 vol.4. Citation on page 21.

HAYKIN, S. S. **Neural networks and learning machines**. 3. ed. Upper Saddle River, NJ: Pearson Education, 2009. Citation on page 15.

HOLLAND, J. H. **Emergence: From Chaos to Order**. [S.l.]: Perseus Publishing, 1999. Citation on page 17.

JEBARA, T.; WANG, J.; CHANG, S.-F. Graph construction and b-matching for semi-supervised learning. In: **Proceedings of the 26th Annual International Conference on Machine Learning**. Montreal, Quebec, Canada: ACM, 2009. p. 441–448. Citations on pages 19 and 20.

JENSEN, H. J. **Self-Organized Criticality: Emergent Complex Behavior in Physical and Biological Systems (Cambridge Lecture Notes in Physics)**. 1. ed. [S.l.]: Cambridge University Press, 1998. Citation on page 17.

JIANG, J.; HUANG, Z.-G.; SEAGER, T. P.; LIN, W.; GREBOGI, C.; HASTINGS, A.; LAI, Y.-C. Predicting tipping points in mutualistic networks through dimension reduction. **Proceedings of the National Academy of Sciences**, n. 18, p. 201714958, 2018. Citation on page 90.

KENNEDY, J. Swarm intelligence. In: ____. **Handbook of Nature-Inspired and Innovative Computing**. Boston, MA: Springer, 2006. p. 187–219. Citations on pages 17 and 21.

KOHONEN, T. The self-organizing map. **Neurocomputing**, v. 21, n. 1, p. 1 – 6, 1998. Citation on page 19.

KOSKO, B. Stochastic competitive learning. **IEEE Transactions on Neural Networks**, v. 2, n. 5, p. 522–529, 1991. Citation on page 19.

KRAMER, O. **Genetic Algorithm Essentials**. Cham: Springer International Publishing, 2017. (Studies in Computational Intelligence, v. 679). Citation on page 17.

LEVER, J. J.; NES, E. H. van; SCHEFFER, M.; BASCOMPTE, J. The sudden collapse of pollinator communities. **Ecology Letters**, v. 17, n. 3, p. 350–359, 2014. Citation on page 90.

LIU, W.; JIANG, X.; PELLEGRINI, M.; WANG, X. Discovering communities in complex networks by edge label propagation. **Scientific Reports**, v. 6, n. 1, p. 22470, 2016. Citation on page 89.

LOOG, M.; JENSEN, A. C. Semi-Supervised Nearest Mean Classification Through a Constrained Log-Likelihood. **IEEE Transactions on Neural Networks and Learning Systems**, v. 26, n. 5, p. 995–1006, 2015. Citation on page 16.

MA, H.; XIE, H.; BROWN, D. Eco-driving assistance system for a manual transmission bus based on machine learning. **IEEE Transactions on Intelligent Transportation Systems**, v. 19, n. 2, p. 572–581, 2018. Citation on page 15.

MERTZ, L. Machine learning takes on health care: Leonard d'avolio's cyft employs big data to benefit patients and providers. **IEEE Pulse**, v. 9, n. 1, p. 10–11, 2018. Citation on page 15.

MORARESCU, I. C.; GIRARD, A. Opinion dynamics with decaying confidence: Application to community detection in graphs. **IEEE Transactions on Automatic Control**, v. 56, n. 8, p. 1862–1873, 2011. Citation on page 18.

MOUSSAID, M.; GARNIER, S.; THERAULAZ, G.; HELBING, D. Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds. **Topics in Cognitive Science**, v. 1, n. 3, p. 469–497, 2009. Citation on page 17.

MOWRER, O. H. **Learning theory and the symbolic processes.** Hoboken, NJ, US: John Wiley & Sons, Inc, 1960. Citation on page 15.

NAGAI, K. H.; SUMINO, Y.; MONTAGNE, R.; ARANSON, I. S.; CHATÉ, H. Collective motion of self-propelled particles with memory. **Phys. Rev. Lett.**, American Physical Society, v. 114, p. 168001, 2015. Citation on page 21.

NETO, F. A.; ZHAO, L. High level data classification based on network entropy. In: **The 2013 International Joint Conference on Neural Networks**. Dallas, TX: IEEE, 2013. v. 1, p. 1–5. Citation on page 19.

NEWMAN, M.; BARABÁSI, A.-L.; WATTS, D. J. **The Structure and Dynamics of Networks: Princeton Studies in Complexity**. [S.l.]: Princeton University Press, 2006. 592 p. Citations on pages 17 and 18.

NEWMAN, M. E. J. **Networks: An Introduction**. 1. ed. New York, NY: Oxford University Press, 2010. Citation on page 18.

NIGAM, K.; MCCALLUM, A. K.; THRUN, S.; MITCHELL, T. Text classification from labeled and unlabeled documents using EM. **Machine Learning**, v. 39, n. 2-3, p. 103–134, 2000. Citation on page 16.

PEMANTLE, R. A survey of random processes with reinforcement. **Probability Surveys**, v. 4, p. 1–79, 2007. Citation on page 89.

PFEIFER, R.; LUNGARELLA, M.; IIDA, F. Self-organization, embodiment, and biologically inspired robotics. **Science**, American Association for the Advancement of Science, v. 318, n. 5853, p. 1088–1093, 2007. Citation on page 18.

QUILES, M. G.; ZHAO, L.; ALONSO, R. L.; ROMERO, R. A. F. Particle competition for complex network community detection. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, v. 18, n. 3, p. 033107, 2008. Citations on pages 18 and 19.

REYNEN, A.; AUDET, P. Supervised machine learning on a network scale: application to seismic event classification and detection. **Geophysical Journal International**, v. 210, n. 3, p. 1394–1409, 2017. Citation on page 15.

SAU, A.; BHAKTA, I. Predicting anxiety and depression in elderly patients using machine learning technology. **Healthcare Technology Letters**, v. 4, n. 6, p. 238–243, 2017. Citation on page 15.

SILVA, T. C.; ZHAO, L. Network-Based High Level Data Classification. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 6, p. 954–970, 2012. Citation on page 19.

____. Network-based stochastic semisupervised learning. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 3, p. 451–466, 2012. Citations on pages 18 and 19.

____. Semi-supervised learning guided by the modularity measure in complex networks. **Neurocomputing**, v. 78, n. 1, p. 30–37, 2012. Citation on page 17.

____. **Machine Learning in Complex Networks**. 1. ed. [S.l.]: Springer, 2016. 331 p. Citations on pages 18, 19, and 20.

TENENBAUM, J. B. A Global Geometric Framework for Nonlinear Dimensionality Reduction. **Science**, v. 290, n. 5500, p. 2319–2323, 2000. Citation on page 20.

UPADHYAYA, S. R. Parallel approaches to machine learning—a comprehensive survey. **Journal of Parallel and Distributed Computing**, v. 73, n. 3, p. 284 – 292, 2013. Models and Algorithms for High-Performance Distributed Data Mining. Citation on page 21.

URIO, P. R.; VERRI, F. A. N.; ZHAO, L. Semi-Supervised Classification by Particle Competition in Complex Network's Edges. **International Journal of Pattern Recognition and Artificial Intelligence**, v. 30, n. 09, p. 1660006, 2016. Citation on page 18.

VAPNIK, V. N. **Statitical learning theory**. New York, NY: Wiley, 1998. Citation on page 17.

WAGSTAFF, K.; CARDIE, C.; ROGERS, S.; SCHROEDL, S. Constrained k-means clustering with background knowledge. In: **Proc. of the 18th International Conference on Machine Learning**. [S.l.: s.n.], 2001. p. 577–584. Citation on page 16.

ZHANG, K.; LAN, L.; KWOK, J. T.; VUCETIC, S.; PARVIN, B. Scaling Up Graph-Based Semisupervised Learning via Prototype Vector Machines. **IEEE Transactions on Neural Networks and Learning Systems**, v. 26, n. 3, p. 444–457, 2015. Citations on pages 17 and 18.

ZHAO, L. Chaotic synchronization for scene segmentation. **International Journal of Modern Physics B**, v. 17, n. 22n24, p. 4387–4394, 2003. Citation on page 18.

ZHAO, L.; MACAU, E. E. N. A network of dynamically coupled chaotic maps for scene segmentation. **IEEE Transactions on Neural Networks**, v. 12, n. 6, p. 1375–1385, 2001. Citation on page 18.

ZHOU, Z.-H.; LI, M. Tri-training: exploiting unlabeled data using three classifiers. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 11, p. 1529–1541, 2005. Citation on page 16.

ZHU, X. **Semi-Supervised Learning Literature Survey**. [S.l.], 2005. Available: <http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf>. Citations on pages 17, 18, and 20.

ZHU, X.; GOLDBERG, A. B.; KHOT, T. Some new directions in graph-based semi-supervised learning. **Proceedings of the IEEE International Conference on Multimedia and Expo**, p. 1504–1507, 2009. Citation on page 18.


ICMC
USP
SÃO CARLOS