Semi-Supervised Learning in Complex Networks

Thiago C. Silva[1] **and Liang Zhao**[1]

[1]Department of Computer Sciences, Institute of Mathematics and Computer Science (ICMC), University of São Paulo (USP)

{thiagoch, zhao}@icmc.usp.br

Abstract. *Semi-Supervised Learning (SSL) is a machine learning scheme which is able to employ both labeled and unlabeled samples in the training process. In this paper, we propose a semi-supervised classification model based on a combined random-deterministic walk of particles in the network (graph) constructed from the input data set. The particles of the same class cooperate among them, while the particles of different classes compete with each other to propagate class labels to the whole network. A rigorous definition of the model is provided. An interesting feature of the proposed model is that each particle only visits a portion of nodes potentially belonging to it due to the competition mechanism. Thus, many long range, apparently meaningless visits are avoided. As a result, the proposed model can achieve a good classification rate while exhibiting low computational complexity order in comparison to other network-based semi-supervised algorithms. Computer simulations carried out for synthetic and real-world data sets show good performance of the model.*

1. Introduction

Nowadays, information reaches us at a remarkable speed and the amount of data it brings is unprecedented. In many situations only a small subset of data items can be effectively labeled. This is because the labeling process is often expensive, time consuming, and requires intensive human involvement. As a result, partially labeled data sets become more frequently encountered. Traditional classifiers are constructed by supervised learning, where only labeled data are considered in the training process and unlabeled examples are simply ignored. On the other hand, there is no mechanism for unsupervised learning, such as data clustering, to treat label information. In order to get a better characterization of partially labeled data sets, semi-supervised classifiers are designed to learn from both labeled and unlabeled data. It turned out to be a new topic of machine learning research that has received increasing attention in the past years [Chapelle et al. 2006].

Semi-supervised methods include generative models [Fujino et al. 2005], cluster-and-label techniques [Dara et al. 2002], co-training and tri-training techniques [Mitchell 1999], low-density separation models, like Transductive Support Vector Machines (TSVM) [Vapnik 2008], and graph-based methods. Many SSL techniques, such as TSVM, can identify data classes of well-defined forms, but usually fail to identify classes of irregular forms. Thus, assumptions on class distributions have to be made. Unfortunately, such information is usually unknown a priori. In order to overcome this problem, graph based methods have been developed over the last years, like Local and Global Consistency [Zhou et al. 2004], Local Learning Regularization [Wu and Schölkopf 2007], Local and Global Regularization [Wang et al. 2008]. The main advantage of graph-based methods is the ability of identifying classes of arbitrary distributions. However, most of the graph-based methods share the regularization framework, differing only in the particular choice of the loss function and the regularizer [Belkin et al. 2005], and most of them have cubic order of computational complexity ($\mathrm{O}(n^3)$). This factor makes their applicability limited to small or middle-sized data sets [Zhu 2005]. As data sets get larger and larger, the development of efficient semi-supervised learning methods is still necessary.

Competition is a natural process observed in nature and in social systems sharing limited resources. Competitive learning is an important category of machine learning and is widely implemented in artificial neural networks to realize unsupervised learning [Kohonen 1990, Jain et al. 2010]. Without a doubt, competitive learning neural networks represent one of the main successes of neural network development. However, at least two problems remain: 1) The constructed network is usually small. So competition occurs among a small number of neurons. Consequently, the model may not exhibit high robustness in data processing. 2) There is not a direct connection between the input data and the trained competitive learning neural networks. When a large data set is mapped to a network of a small number of neurons, it becomes hard to see the correspondence between the original data and the trained neural networks. This is one of the reasons why neural networks sometimes are considered as "black box" systems.

In order to heritage the interesting features and at the same time overcome the problems of competitive learning neural networks, Quiles et. al. [Quiles et al. 2008] proposed a particle walking model to realize the competitive learning mechanism. In this paper, we propose a network-based semi-supervised learning model based on particle competition and cooperation in the network constructed from the input data set. This model not only maintains the competition mechanism presented in Ref. [Quiles et al. 2008], but also introduces a cooperative mechanism. In this way, particles of the same class proceed in the network in a cooperative manner to propagate their labels, while particles of different classes compete with each other to determine the class borders. Another interesting feature of the proposed technique is that it has a local label spreading fashion, i.e., at each time step, each particle spreads its label to a neighbor node chosen by the combined random-deterministic rule. Due to the competition mechanism, each particle only visits a portion of nodes potentially belonging to the current particle or its teammates, while it is not allowed to visit those nodes definitely occupied by other teams of particles. It can be roughly understood that our method has a "divide-and-conquer" effect embedded in the competition-cooperation scheme. In this way, many long-range redundant operations are avoided. As a result, the proposed method has a lower computational complexity order. Since the underlying network is constructed directly from the input data set, the correspondence between the input data and the processing result (the final network) is maintained. As a result, the "black box" effect can be avoided at a large extent.

The remainder of the paper is organized as follows. The proposed model definition is described in Sect. 2. In Sect. 3, computer simulations are performed to show how the proposed model solves data classification by using artificial and real data sets. Finally, Sect. 4 concludes the paper.

2. Model Description

Consider that we are given a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges. In the competitive learning model, a set of particles $\mathcal{K} = \{1, ..., K\}$ is inserted into the vertices of the network in a random manner. Each particle can be thought of carrying a flag and its objective is to conquer new territories - represented here by the vertices - while defending its owned territories. In this case, a competition process will naturally take place amongst the particles. When a particle visits an arbitrary vertex, it strengthens its own domination level on the vertex and simultaneously weakens the domination levels of all other rival particles on the same vertex. It is expected to this model, in a broad horizon of time, to end up uncovering the classes in the network in such a way that each particle (or a team of particle through cooperation) dominates a class. In the next sections, we will give detailed explanation on how to derive the competitive dynamical system with the above-mentioned characteristics.

2.1. The Competitive Transition Matrix

Regarding the movement policy of each particle $j \in \mathcal{K}$, it is basically composed of two distinct types: 1) a random movement term, modeled by the matrix $\mathbb{P}^{(j)}_{\text{rand}}$, which permits the particle to adventure through the network, without accounting for the defense of the previously dominated vertices; 2) a deterministic movement term, modeled by the matrix $\mathbb{P}^{(j)}_{\text{det}}$, which is responsible for inducing the particle to reinforce the vertices that are owned by itself, i.e., the particle will prefer visiting its dominated vertices, instead of a randomly selected one. In order to model such dynamics, consider that $p(t) = [p^{(1)}(t), p^{(2)}(t), ..., p^{(K)}(t)]$ denotes the localization of the set of K particles presented to the network, where the jth-entry, $p^{(j)}(t)$, indicates the location of the particle j in the network at time t, i.e., $p^{(j)}(t) \in \mathcal{V}, \forall j \in \mathcal{K}$. It is desired to find a transition matrix that governs the probability distribution of the particles' movement to the immediate future state, $p(t+1) = [p^{(1)}(t+1), p^{(2)}(t+1), ..., p^{(K)}(t+1)]$.

With only those two types of movement behavior, it is possible that the owned territory of each particle to be swapped between them. As there is no force that compels the particles to regress to their owned territory time to time, the particles can be considered free to travel anywhere in the network with no penalties; so, the aforementioned scenario could happen a substantial number of times until the system comes to a stationary state. On account of that, the number of steps that the system would require to converge is expected to be usually high with only these two movement behaviors. In order to overcome that, we introduce energy levels for all particles and, with the aid of this measure, we apply some restrictions on all particles in the following manner: if a particle visits a vertex that is being dominated by itself, then the corresponding energy of that particle increases. Likewise, if a particle visits a vertex that is being dominated by a rival particle, then the corresponding energy of that particle is drained. If the actual energy of a specific particle reaches a certain minimum threshold, then it is said that the particle has died at that step. In the subsequent step, that particle is automatically resurrected in a vertex that belongs to it in a random manner. With this behavior, we expect that the particles will no longer wander free in the network, possibly swapping territories with other particles several times. Thus, this characteristic is expected to restrain the particles' effective acting region. In the semi-supervised version, which we will further detail in this section, it is guaranteed that there will be at least one vertex that is being dominated by each particle, so that the behavior of teleporting back to its territory can always be applied.

With the intent of modeling such dynamics, we introduce the following random quantity $S^{(t)} = [S^{(1)}(t), \ldots, S^{(K)}(t)]$, where the jth-entry, $S^{(j)}(t) \in \{0, 1\}$, indicates whether the particle j is dead or alive at time t. Specifically, if $S^{(j)}(t) = 1$, then particle j is said to be dead. Likewise, when $S^{(j)}(t) = 0$, the particle is said to be alive. Thus, if $S^{(j)}(t) = 0$, the particle navigates in the network according to a combined behavior of randomness and determinism. However, if $S^{(j)}(t) = 1$, the particle switches its movement policy to a new transition matrix, here entitled $\mathbb{P}^{(j)}_{\text{res}}(t)$, which is responsible for taking the particle back to its owned territory ("safe ground"). In brief terms, $S(t)$ acts as a switch that determines the movement policy of all particles at time t. With all this information in mind, we are able to define the transition matrix associated to the particle j as:

$$\mathbb{P}^{(j)}_{\text{transition}}(t) \triangleq (1 - S^{(j)}(t)) \left[\lambda \mathbb{P}^{(j)}_{\text{det}}(t) + (1 - \lambda)\mathbb{P}^{(j)}_{\text{rand}}(t)\right] + S^{(j)}(t)\mathbb{P}^{(j)}_{\text{res}}(t) \qquad (1)$$

where $\lambda \in [0, 1]$ indicates the desired fraction of deterministic movement that all particles in the network will perform, $\mathbb{P}^{(j)}_{\text{det}}(t)$ portrays the transition matrix with a probability distribution according to the deterministic behavior described above and, likewise, $\mathbb{P}^{(j)}_{\text{rand}}(t)$ describes the random behavior, $S^{(j)}(t)$ indicates whether particle j is alive or dead, and $\mathbb{P}^{(j)}_{\text{res}}(t)$ is responsible for the particle resurrection behavior. It is worth noting that Eq. (1) is a convex combination of two transition matrices (the first term is itself a combination of two transition matrices, too), since the sum of the coefficients is unitary, therefore, the resulting matrix is guaranteed to be another transition matrix. Now we proceed to define each matrix that appears in Eq. (1) in a detailed manner.

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

$$\mathbb{P}^{(j)}_{\text{rand}}(i, k) \triangleq \frac{a_{i,k}}{\sum_{u=1}^{V} a_{i,u}} \qquad (2)$$

where $a_{i,k}$ denotes the (i, k)th-entry of the adjacency matrix A of the graph. Note that Eq. (2) resembles the traditional Markovian matrix for a single random walker, here symbolized as a particle. Also note that matrix $\mathbb{P}^{(j)}_{\text{rand}}(t)$ is time-invariant and is the same for every particle in the network, therefore, we will drop the independent variable t and the superscript j whenever the situation makes it clear. In short terms, the probability of an adjacent neighbor to be visited using only the random movement behavior is proportional to the edge weight linking the vertex that a specific particle is visiting and that neighbor vertex.

In order to assist in the calculation of the matrix associated to the deterministic movement term, $\mathbb{P}^{(j)}_{\text{det}}(t)$, for a given particle $j \in \mathcal{K}$, we introduce the following random quantity: $N_i(t) \triangleq [N_i^{(1)}(t), N_i^{(2)}(t), ..., N_i^{(K)}(t)]$, where $\dim(N_i(t)) = 1 \times K$ and $N_i(t)$

stands for the absolute number of visits that vertex i has received until the time t, including t, by all the particles scattered throughout the network. Specifically, the jth-entry, $N_i^{(j)}(t)$, indicates the number of visits made by the particle j to vertex i up to time t. We now simply extend this notation to all vertices in the network, defining the global matrix that maintains the number of visits made by every particle in the network to all the vertices as: $N(t) \triangleq [N_1, N_2, \dots, N_V]^T$, where $\dim(N(t)) = V \times K$.

Let us also formally define the domination level vector of vertex i, $\bar{N}_i(t)$, according to the following random variable: $\bar{N}_i(t) \triangleq [\bar{N}_i^{(1)}(t), \bar{N}_i^{(2)}(t), ..., \bar{N}_i^{(K)}(t)]$, where $\dim(\bar{N}_i(t)) = 1 \times K$ and $\bar{N}_i(t)$ denotes the relative frequency of visits of all particles in the network to vertex i until the time t, including t. Particularly, the jth-entry, $\bar{N}_i^{(j)}(t)$, indicates the relative frequency of visits performed by particle j to vertex i up to time t. Similarly to the previous case, we extend this notion to all vertices in the network, defining the domination level matrix that sustains all the domination levels imposed by every particle in the network to all the vertices as: $\bar{N}(t) \triangleq [\bar{N}_1, \bar{N}_2, \dots, \bar{N}_V]^T$, where $\dim(N(t)) = V \times K$. Mathematically, we define each entry of $\bar{N}_i^{(j)}(t)$ as:

$$\bar{N}_i^{(j)}(t) \triangleq \frac{N_i^{(j)}(t)}{\sum_{p=1}^{K} N_i^{(p)}(t)} \tag{3}$$

In view of that, we can define the (i,k)th-entry of the matrix responsible for the deterministic movement behavior of a single particle $j \in \mathcal{K}$, denoted here by $\mathbb{P}_{\text{det}}^{(j)}(t)$, at time t, as following:

$$\mathbb{P}_{\text{det}}^{(j)}(i,k,t) \triangleq \frac{a_{i,k}\bar{N}_k^{(j)}(t)}{\sum_{u=1}^{V} a_{i,u}\bar{N}_u^{(j)}(t)} \tag{4}$$

Clearly, from Eq. (4), it can be observed that each particle has a different transition matrix associated to its deterministic movement and that, unlike the matrix associated to the random movement, this matrix is time-variant with dependence on the domination levels of all the vertices ($\bar{N}(t)$) in the network at the time t. It is worth remarking that the approach taken here to characterize the deterministic movement of the particles is the visiting frequency of each particle to a specific vertex, in such a way that as more visits are performed from a specific particle to an arbitrary vertex, the higher the chance will be of the same particle to come back visiting the same vertex again and again. Also it is important to emphasize that Eq. (4) produces two distinct features presented by a natural competition model: 1) the strengthening of the domination level of the visiting particle to a vertex; 2) the consequent weakening of the domination levels of all other particles on that same vertex. This behavior is naturally represented by the model due to the frequency approach taken.

Now we define each entry of $\mathbb{P}_{\text{res}}^{(j)}(t)$ that is accounted for teleporting a dead particle $j \in \mathcal{K}$ back to its owned territory as:

$$\mathbb{P}_{\text{res}}^{(j)}(i,k,t) \triangleq \frac{\mathbb{1}_{\left\{\underset{m\in\mathcal{K}}{\max}\left(\bar{N}_{p^{(j)}(t)}^{(m)}(t)\right)=j\right\}}\bar{N}_k^{(j)}(t)}{\sum_{u=0}^{V}\mathbb{1}_{\left\{\underset{m\in\mathcal{K}}{\max}\left(\bar{N}_u^{(m)}(t)\right)=j\right\}}\bar{N}_u^{(j)}(t)} \tag{5}$$

where $\underset{m\in\mathcal{K}}{\max}(.)$ returns the index m which maximizes the argument and $\mathbb{1}_{\{.\}}$ is the indicator functions that yields 1 if the argument is logically true and 0, otherwise. Indeed, a careful analysis of the expression in Eq. (5) shows that the denominator term sums over all dominations levels of all the vertices that are being dominated by particle j, i.e., vertices that are dominated by rival particles do not have their values considered. With that in mind, Eq. (5) only results in non-zero transition probabilities for vertices k that are being dominated by particle j, regardless of the existence of a connection between i and k in the adjacency matrix. In essence, once the particle is dead, the switch is enabled, which in turn compels the particle j to return to its previously owned territory, no matter if there is a physical connection or not in the adjacency matrix.

Now we proceed to the development of the particle's energy update rule. Firstly, it is useful to introduce the random variable $E(t) = [E^{(1)}(t), \dots, E^{(K)}(t)]$, where the jth-entry, $E^{(j)}(t) \in [\omega_{\min}, \omega_{\max}]$, $\omega_{\max} \geq \omega_{\min}$, denotes the energy level of particle j at time t, whose update rule is given by:

$$E^{(j)}(t) = \begin{cases} \min(\omega_{\max}, E^{(j)}(t-1)+\Delta), & \text{if } \text{owner}(j,t) \\ \max(\omega_{\min}, E^{(j)}(t-1)-\Delta), & \text{if } \neg\ \text{owner}(j,t) \end{cases} \tag{6}$$

where $\text{owner(j, t)} = \left(\underset{m\in\mathcal{K}}{\max}\left(\bar{N}_{p^{(j)}(t)}^{(m)}(t)\right) = j\right)$ is a logical expression that essentially yields true if the vertex that particle j visits at time t (vertex $p^{(j)}(t)$) is being dominated by the visiting particle, and false otherwise; $\dim(E(t)) = 1 \times K$; $\Delta > 0$ symbolizes the increment or decrement of energy that each particle will receive at time t. Indeed, the first expression in Eq. (6) represents the increment of the particle's energy and occurs when the particle j visits a vertex $p^{(j)}(t)$ which is dominated by itself, i.e., $\underset{m\in\mathcal{K}}{\max}\left(\bar{N}_{p^{(j)}(t)}^{(m)}(t)\right) = j$. Similarly, the second expression in Eq. (6) portrays the decrement of the particle's energy and occurs when particle j visits a vertex $p^{(j)}(t)$ which is not dominated by itself, i.e., there is a domination level on that vertex that is higher than the one imposed by particle j. Hence, in this model, particles will be given a penalty if they are wandering in rival territory, so as to minimize aimless navigation of the particles in the network which would only reduce the speed of convergence of the dynamical system. By the same reasons, we expect this behavior to improve the final classification rate of the algorithm.

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

$$S^{(j)}(t) = \mathbb{1}_{\{E^{(j)}(t)=\omega_{\min}\}} \tag{7}$$

where $\dim(S(t)) = 1 \times K$. Specifically, $S^{(j)}(t) = 1$ if $E^{(j)}(t) = \omega_{\min}$ and 0, otherwise. As there is an upper limit for the random variable $E^{(j)}(t)$, it is clear that if particle j frequently visits vertices owned by rival particles, its energy will decrease in such a way that it could reach the minimum energy $\omega_{\min}$ and, hence, die. The upper limit, $\omega_{\max}$, was established to prevent any particle in the network to keep increasing its energy to an undesirably high value (by constantly visiting vertices in its territory), and, once this energy is high enough, it could go far away from its territory and visit a substantial number of vertices belonging to rival particles before dying, thus, considerably decreasing the convergence time and classification rate of the dynamical system.

Having defined each matrix associated to each particle, we couple all these matrices into a representative transition matrix $\mathbb{P}_{\text{transition}}(t)$ using the following fact: when a specific particle is alive, the movement of each particle is independent of all the other rival particles, because at each time step the next vertex to be visited by a specific particle depends solely on a convex combination of randomness - directly proportional to the edge weight of the neighbor vertices, see Eq. (2) - and determinism - in this case, is a function of the domination levels of all the vertices in the neighborhood, see Eq. (4). Due to that, the location of the other particles causes no influence over the action of choosing the next vertex to be visited by a given particle. The same idea can be applied when a specific particle is dead. In virtue of this property, we can describe the full matrix associated to the transition $p(t)$ to $p(t+1)$, i.e., for all particles in the network, as:

$$\mathbb{P}_{\text{transition}}(t) = \mathbb{P}^{(1)}_{\text{transition}}(t) \otimes \ldots \otimes \mathbb{P}^{(K)}_{\text{transition}}(t) \tag{8}$$

where $\otimes$ denotes the Kronecker tensor product operator. In this way, Eq. (8) completely specifies the transition distribution matrix for *all* the particles in the network and not only one particle as is expressed in Eq. (1).

Essentially, $p(t+1)$ can be seen as a discrete stochastic process whose probability distribution is given by the row indicated by the scalar form of $p(t)$ (to be defined) of matrix $\mathbb{P}_{\text{transition}}(t)$, which strictly depends on $N(t)$ for its construction. Upon doing so, we need to enumerate the states of the particle localization $p(t)$ such as to be feasible the usage of the transition matrix, because, for $K \geq 2$, $p(t)$ will be a vector and we would no longer be able to conventionally define row $p(t)$ of $\mathbb{P}_{\text{transition}}(t)$. For that, we simply enumerate the state $p(t) = [p^{(1)}(t), p^{(2)}(t), ..., p^{(K)}(t)]$ to a scalar form, respecting the natural ordering of the tuples, i.e., $p(t) = [1, 1, ..., 1, 1]$ (all particles at vertex 1) denotes the first state; $p(t) = [1, 1, ..., 1, 2]$ (all particles at vertex 1, except the last particle, which is at vertex 2) is the second state; and so on, up to the scalar state V^K.

2.2. The Semi-Supervised Competitive Learning Model

In light of all we have obtained in the previous section, we are ready to enunciate the proposed dynamical system which models the competition of particles in a given network. The internal state of the dynamical system has been chosen to be: $X(t) = [N(t)\ \ p(t)\ \ E(t)\ \ S(t)]^T$ and the proposed competitive dynamical system is given by:

$$\phi : \begin{cases} N_i^{(j)}(t+1) & = N_i^{(j)}(t) + \mathbb{1}_{\{p^{(j)}(t+1)=i\}} \\ E^{(j)}(t+1) & = \begin{cases} \min(\omega_{\max}, E^{(j)}(t) + \Delta), \text{if owner}(j,t) \\ \max(\omega_{\min}, E^{(j)}(t) - \Delta), \text{if } \neg \text{ owner}(j,t) \end{cases} \\ S^{(j)}(t+1) & = \mathbb{1}_{\{E^{(j)}(t+1)=\omega_{\min}\}} \end{cases} \tag{9}$$

where, by the considerations that we have previously stated, $\dim(N(t)) = V \times K$, $\dim(p(t)) = 1 \times K$, $\dim(E(t)) = 1 \times K$, and $\dim(S(t)) = 1 \times K$, resulting that $\dim(X(t)) = (V+3) \times K$, with $N_i^{(j)}(t) \in [1, \infty), (i,j) \in \mathbb{S}$, where $\mathbb{S}$ is the space spawned by $\mathcal{V} \times \mathcal{K}$. Observe that $p(t+1)$ has no closed form because it is qualified as a distribution with dependence on $p(t)$ and $N(t)$, therefore its acquisition is merely by random number generation. Succinctly, the internal state of system ϕ, $X(t)$, carries the current total number of visits made by each particle to each vertex in the network, the current localization of all particles in the network, the current energy that each particle holds, and the information about each particle whether it is current alive or dead. The first equation of system ϕ is responsible for updating the number of visits at vertex i by particle j up to time t; the second equation is used to maintain the current energy levels of all the particles inserted in the network; and the third equation is used to trigger the particle dead or alive, depending on its actual energy level. It is valuable to emphasize that the first expression of system ϕ must be used for every $(i,j) \in \mathbb{S}$ and the second and third expressions must be performed for every $j \in \mathcal{K}$ with the intention of one properly derive the full state $X(t)$ of the system ϕ.

2.3. The Initial Conditions of the System

In order to run system ϕ, we need a set of initial conditions. Firstly, the initial position $p(0)$ of the particles is user-controllable. Additionally, consider a set of classes $\mathcal{C}$ and a set of pre-labeled examples $\mathcal{V}_{\text{L}} \subset \mathcal{V}$. Let $\mathcal{L}$ denote the set that indicates the pair of a labeled node and its corresponding class, i.e., $\mathcal{L} = \{(v_1, c_1), ..., (v_{|\mathcal{V}_{\text{L}}|}, c_{|\mathcal{V}_{\text{L}}|})\}$, where $v_i \in \mathcal{V}_{\text{L}}$, and $c_i \in \mathcal{C}$, $0 \leq i \leq |\mathcal{L}| = |\mathcal{V}_{\text{L}}|$. Then, we have that each entry of $N(0)$ is given by:

$$N_i^{(j)}(0) = \begin{cases} \infty, & \text{if particle j represents node i} \\ 1 + \mathbb{1}_{\{p^{(j)}(0)=i\}}, & \text{otherwise} \end{cases} \tag{10}$$

where we apply Eq. (10) to every $(i,j) \in \mathbb{S}$. Note that the scalar 1 was used in the second expression of Eq. (10) in order to unlabeled and not visited vertices at time 0 to have their calculation well-defined, according to Eq. (3). Regarding the initial condition of $E(0)$, we desire a fair competition amongst the particles, so we place isonomy in their initial energy values, i.e., all particles $j \in \mathcal{K}$ start out with the same energy level given by:

$$E^{(j)}(0) = \omega_{\min} + \left(\frac{\omega_{\max} - \omega_{\min}}{K}\right) \tag{11}$$

Lastly, the variable that accounts for indicating whether the particle j is dead or alive at the initial step, $S^{(j)}(0)$, $\forall j \in \mathcal{K}$, is given by $S^{(j)}(0) = 0$, i.e., we deliberately set alive all particles in the network in the beginning of the process.

3. Computer Simulations

In this section, we present simulation results in order to show the effectiveness of the proposed competitive model. Specifically, in Subsect. 3.1 we analysis the behavior of the dynamical system ϕ by using a simple artificial network; in Subsect. 3.2, we provide results of semi-supervised learning on real data sets as well as a comparison against several well-known semi-supervised learning techniques.

3.1. Simulations on Synthetic Data Sets

In order to facilitate the understanding of how the proposed technique works, we design a synthetic data set of two simple classes, each of which with 50 vertices. We have inserted into the network 2 particles, each one representing a class. With this simple data set, we are able to closely observe the behavior of our proposed algorithm working on synthetic data. Figure 1a shows the initial configuration of the network, where the colored dots symbolize labeled samples. The black dots denote unlabeled data. The ownership of any vertex is given by the particle which has the highest domination level over that vertex. According to Eq. (10), the vertices that are initially labeled have their ownership fixed to its corresponding representative particle. For this simulation, we fix $\lambda = 0.6$, $\Delta = 0.05$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. As the dynamical system evolves, the particles will visit the vertices of the network in agreement with the probability distribution given by the matrix $\mathbb{P}_{\text{transition}}(t)$. Figure 1b shows the ownership of each vertex after 300 iterations, Fig. 1c depicts the ownership state after 600 iterations, and Fig. 1d reveals the stationary ownership state of the algorithm $\bar{N}(t)$, which is reached after 3000 iterations. We now take a closer look at the evolutional behavior of the average domination level of the vertices that belong to the same class. Figure 2a provides the average domination level imposed by the particle representing the initially blue labeled vertex on the vertices 1 to 50 (blue class) and 51 to 100 (red class), whereas Fig. 2b displays the same information for the particle representing the initially red labeled vertex. Clearly, as time progresses, one can see that the classes are unmistakeably separated by the competitive system.

3.2. Simulations on Benchmark Data Sets

In order to measure the performance of the proposed method, we have applied it to 7 standard semi-supervised data sets. For a detailed description of the data set, refer to [Chapelle et al. 2006]. For each data set in the benchmark, it is provided 12 benchmark partitions,

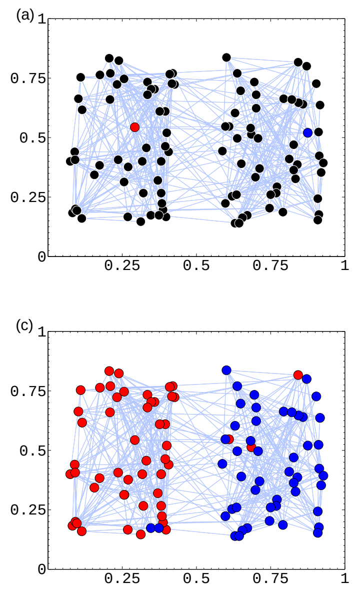
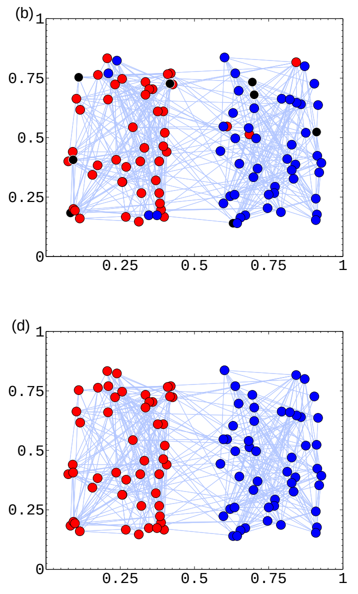
Figure 1. Illustration of an artificial classification process through competitive particle walking. The total number of nodes is $V = 100$. (a) A snapshot of the initial configuration: there are two previously labeled nodes (red and blue nodes) and $K = 2$ particles, each one representing a labeled node. The black dots represent unlabeled data. The particles have been spawned at their representative labeled node. (b) A snapshot at iteration 300. (c) A snapshot at iteration 600. (d) A snapshot at iteration 3000.

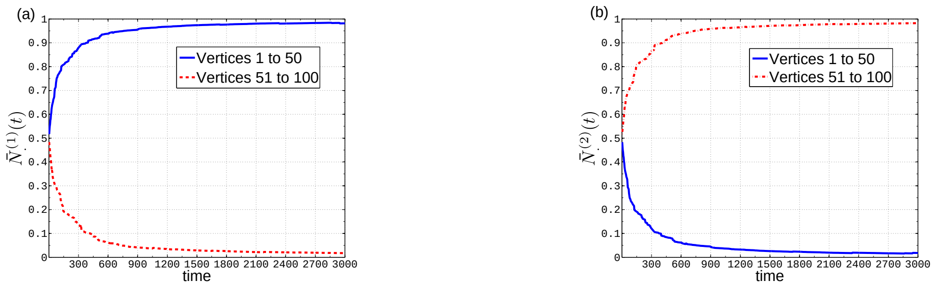

Figure 2. Evolutional behavior of the average class domination level imposed by the particles in the network. (a) Average class domination level imposed by particle 1. (b) Average class domination level imposed by particle 2.

each of which having 10 distinct non-biased labeled vertices. In this partition composition, it is ensured that exists at least 1 labeled node for each class in the database. For comparison matters, we have conducted experiments against a selected set of semi-supervised techniques, whose simulations results were performed under optimal conditions and were integrally taken from [Chapelle et al. 2006]. For comparison purposes, we have used the kNN graph formation technique with optimized k in the interval $1 \leq k < 30$ and have also optimized the algorithm parameters in the intervals: $0 \leq \lambda \leq 1$, $0 < \Delta < 1$, $0 < \omega_{\max} < 5$ with $\omega_{\min} = 0$ fixed. The optimization is realized by using the genetic algorithm available in the Global Optimization Toolbox of MATLAB with its default pa-

Table 1. Test errors (%) with 10 labeled training points and the corresponding average rank of each technique.

Technique	g241c	g241d	Digit1	USPS	COIL	BCI	Text	Avg. Rank
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12	**9.86**
Support Vector Machines	47.32	46.66	30.60	20.03	68.36	49.85	45.37	**14.14**
Maximum Variance Unfolding	47.15	45.56	14.42	23.34	62.62	47.95	45.32	**9.86**
Laplacian Eigenmaps	44.05	43.22	23.47	19.82	65.91	48.74	39.44	**10.00**
Quadratic Criterion and Class Mass Regularization	39.96	46.55	9.80	13.61	59.63	50.36	40.79	**7.86**
Discrete Regularization	49.59	49.05	12.64	16.07	63.38	49.51	40.37	**10.86**
Transductive Support Vector Machines	24.71	50.08	17.77	25.20	67.50	49.15	31.21	**10.86**
Spectral Graph Transducer	22.76	18.64	8.92	25.36	-	49.59	29.02	**6.50**
Cluster Kernels	48.28	42.05	18.73	19.41	67.32	48.31	42.72	**10.86**
Data-Dependent Regularization	41.25	45.89	12.49	17.96	63.65	50.21	-	**9.83**
Low-Density Separation	28.85	50.63	15.63	17.57	61.90	49.27	27.15	**8.43**
Laplacian Regularized Least Squares	43.95	45.68	5.44	18.99	54.54	48.97	33.68	**6.14**
Conditional Harmonic Mixing	39.03	43.01	14.86	20.53	-	46.90	-	**7.20**
Local and Global Consistency	45.82	44.09	9.89	9.03	63.45	47.09	45.50	**7.29**
Label Propagation	42.61	41.93	11.31	14.83	55.82	46.37	49.53	**5.57**
Linear Neighborhood Propagation	47.82	46.24	8.58	17.87	55.50	47.65	41.06	**7.43**
Proposed Method	43.89	46.47	8.10	15.69	54.18	48.00	34.84	**5.29**

rameters. The values obtained for the proposed method are averaged by 100 realizations on each of the 12 subsets. The results obtained from these techniques against the aforementioned databases are reported in Table 1. From the same table, we can conclude that our technique had a satisfactory result in comparison to the other techniques. With only a few nodes, our technique was able to correctly spread the labels to the vicinity of nodes with the aid of the competitive mechanism provided by the algorithm. This is an attractive characteristic, since the task of vertex labeling is often expensive and cumbersome, which generally involves the work of a human expert. In order to statistically verify whether the algorithms presented in Table 1 have significant difference to each other, we apply the Friedman Test with a significance level of 0.05. See [Demšar 2006] for details. We get that the critical statistical descriptor is lower than the quantity provided by the data itself, therefore, the null hypothesis is rejected and we confirm that the algorithms under analysis present significant difference. Specifically, we can see that our algorithm is superior to the others for the set of databases that we have conducted our tests.

4. Conclusions

This paper proposes a nonlinear and stochastic mathematical model for competitive learning in complex networks, biologically inspired by the competition process taking place in many nature and social systems. In this model, several particles, each of which representing a class, navigate in the network to explore their territory and, at the same time, attempt to defend its dominion against rival particles. If several particles propagate the same class label, then a team is formed, and a cooperation process amongst these particles occurs. A confinement mechanism was proposed in order to prevent the particles to wander free in the network, possibly reducing the overall classification rate of the algorithm and its speed of convergence. Consequently, the proposed technique spreads labels in a local fashion, instead of the traditional semi-supervised techniques which propagates labels in a global fashion. Simulations were carried out with the purpose of quantifying the robustness of the proposed technique on synthetic and real-world data sets for the task of data classification and reasonable results have been obtained. More importantly, this work is an attempt to provide an alternative way to the study of competitive learning.

References

Belkin, M., P., N., and Sindhwani, V. (2005). On manifold regularization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, pages 17–24, New Jersey. Society for Artificial Intelligence and Statistics.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA.

Dara, R., Kremer, S., and Stacey, D. (2002). Clustering unlabeled data with SOMs improves classification of labeled real-world data. In *Proceedings of the World Congress on Computational Intelligence (WCCI)*, pages 2237–2242.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Fujino, A., Ueda, N., and Saito, K. (2005). A hybrid generative/discriminative approach to semi-supervised classifier design. In *AAAI-05, Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 764–769.

Jain, L. C., Lazzerini, B., and (eds.), U. H. (2010). *Innovations in ART Neural Networks (Studies in Fuzziness and Soft Computing)*. Physica-Verlag, Heidelberg.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Mitchell, T. M. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*.

Quiles, M. G., Zhao, L., Alonso, R. L., and Romero, R. A. F. (2008). Particle competition for complex network community detection. *Chaos*, 18(3):033107.

Vapnik, V. N. (2008). *Statistical Learning Theory*. Wiley-Interscience, New York.

Wang, F., Li, T., Wang, G., and Zhang, C. (2008). Semi-supervised classification using local and global regularization. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 726–731. AAAI Press.

Wu, M. and Schölkopf, B. (2007). Transductive classification via local learning regularization. In *11th International Conference on Artificial Intelligence and Statistics*, pages 628–635. Microtome.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, volume 16, pages 321–328. MIT Press.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.