Stochastic Competitive Learning Applied to Handwritten Digit and Letter Clustering

Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao
Department of Computer Sciences, Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
Av. Trabalhador São-carlense, 400, 13560-970, São Carlos, SP, Brazil
e-mail: {thiagoch, thiagohc, zhao}@icmc.usp.br.

***Abstract*—Competitive learning is an important mechanism for data clustering and pattern recognition. In this paper, we present a rigorous definition of a new type of competitive learning scheme realized on large scale networks. In this model, several particles walk in the network and compete with each other to occupy as many nodes as possible, while attempting to reject intruder particles. As a result, each particle will dominate a cluster of the network. Moreover, we propose an efficient method for determining the right number of clusters by using the information generated by the competition process itself, avoiding the calculation of an external evaluating index. In this work, we apply the model to handwritten data clustering. Computer simulations reveal that the proposed technique obtains satisfactory cluster detection accuracy.**

***Keywords*-Stochastic competitive learning; handwritten pattern clustering.**

I. Introduction and Related Work

Competition is a natural process observed in nature and in many social systems sharing limited resources, such as water, food, mates, territory, recognition, etc. Competitive learning is an important category of machine learning and is widely implemented in artificial neural networks to realize unsupervised learning. Early works include the development of the famous Self-Organizing Map (SOM) [1], Differential Competitive Learning [2], and Adaptive Resonance Theory (ART) [3]. From then on, many competitive learning neural networks have been developed [4] and a wide range of applications, such as data clustering, data visualization, pattern recognition, and image processing have been considered [5]. Without a doubt, competitive learning represents one of the main successes of neural network development. However, at least two problems remain: (i) 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. (ii) 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, apparently, it is hard or even impossible to see the correspondence between the original data and the processing result (the trained neural network). This is one of the reasons why neural networks sometimes are considered as "black box" systems.

A random walk is a mathematical formalization of a trajectory that consists of taking successive random steps. It has been used to describe many natural phenomena and has also been applied to solve a wide range of engineering problems [6], such as graph matching and pattern recognition, image segmentation, neural network modeling, network centrality measure, network partition, communication network construction and analysis, among many others. However, to our knowledge, there is no theory yet to describe a process of several interacting random walks.

In order to inherit the interesting features and, at the same time, overcome the problems of competitive learning neural networks, in this paper, we study a new type of competitive learning mechanism. Consider a large scale graph (network), where several particles walk in the network and compete with each other to mark their own territory (occupy as many nodes as possible), while they also attempt to reject particle intruders. Each particle can perform a random walk by choosing any neighbor to visit, a biased walk by choosing the node with the highest domination to visit or a combination of them. The straightforward applications are the community detection in networks and data clustering. In essence, data clustering can be considered as a community detection problem once a network is constructed from the original data set, where each node corresponds to a data item and the connections are established by using a certain similarity measure. Considering such applications, the competitive walking process reaches dynamics equilibrium when each community or a data cluster is dominated by only one particle.

Interestingly, the particle competition process is rather similar to many natural and social processes, such as resource competition by animals, territory exploration by humans (animal), election campaigns, etc. Moreover, the combined random-biased walking of competitive particles can largely improve the community detection rate. In this way, the model corroborates the importance of the randomness role in evolutionary systems, i.e., randomness serves to automatically escape from some undesirable traps and to explore new spaces. Thus, our model shows that a certain level of randomness is essential for the learning process. Such randomness represents the "I don't know" state and serves as a novelty finder. It can also help learning agents, like particles in our model, to escape from traps in physical or learning spaces.

The particle competition model was originally proposed in [7]. However, only a procedure is defined there. In this paper, we present a rigorous model definition, i.e., we represent the particle competition process by a stochastic dynamical system. Besides of this, we also apply the model to solve data clustering problems. Moreover, we have developed an efficient method for determining the right cluster number by using the dominance level information generated by the competition process itself, i.e., the determination procedure is already embedded in the model. As a result, it does not increase the model's complexity order. Since the determination of the right or optimal number of clusters is an important issue in data clustering, our method presents a contribution to this end. Another interesting feature of the model is that 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.

Over the last decade there has been an increasing interest in network research, with the focus shifting away from the analysis of single small graphs to the consideration of large-scale ones, called *complex networks*. Such networks have emerged as a unified representation of complex systems in various branches of science. In general, they are used to model systems having a nontrivial topology and are composed of a large amount of vertices [8]. One of the striking phenomena of complex networks is the presence of *communities*. The notion of *community* in networks is straightforward, each community is defined as a subgraph whose nodes are densely connected within itself but sparsely connected with the rest of the network. Community detection in complex networks has turned out to be an important topic in graph and data mining [9]. In graph theory, community detection corresponds to graph partition, which has been shown to be a NP-complete problem [9]. For this reason, a lot of efforts have been spent to develop more efficient approximate solutions, such as the spectral method [10], modularity optimization [11], among others. For a recent review of this topic, see [9]. As having been mentioned, in this work, we will show how the proposed model can be applied to solve community detection problems and consequently to data clustering problems. Computer simulation results show that the network-based approach has an advantage to detect various forms of cluster.

The remainder of the paper is organized as follows. The proposed model definition is described in Section 2. In Section 3, computer simulations are performed to show how the proposed model solves the problem of finding the number of clusters in the data set, as well as how the algorithm behaves in data clustering tasks. Finally, Section 4 concludes the paper.

II. Model Description

In this section, the proposed competitive learning model pertaining to the unsupervised scheme is presented in details.

A. A Brief Overview of the Model

Consider that we are given a graph $G = \langle V, E \rangle$, where $\mathcal{V} = \{v_1, \dots, v_V\}$ is the set of nodes and $E = \{e_1, \dots, e_E\} \subseteq \mathcal{V} \times \mathcal{V}$ 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, will end up uncovering the clusters in the network in such a way that each particle dominates a cluster.

A particle in this model can be at two modes: either dead or alive. Whenever the particle is alive, it navigates in the network with a combined behavior of randomness and biased walking, whereas, when it is dead, the particle changes to a movement term that compels it to get back to its owned territory. The randomness term is responsible for the adventuring behavior of the particle, i.e., it visits vertices without taking into account their domination levels. The biased walking term is responsible for the defensive behavior of the particle, i.e., it prefers to reinforce its owned territory rather than to visit a vertex that is not being dominated by the particle. In order to make this process suitable, each particle carries an energy term with it. This energy increases when the particle is visiting a vertex whose owner is the visiting particle, and decreases whenever it visits a vertex that is being owned by a rival particle. If this energy drops to a minimum allowed value, the particle dies at that time step and teleports back to safe ground, i.e., its owned territory at the next time step. With this confinement mechanism, we expect to restrain the acting region of each particle and, thus, reduce long range, apparently meaningless visits in the network.

B. The Competitive Transition Matrix

In this section, we supply how each probability matrix that composes the particle movement is constructed.

1) Determination of $\mathbb{P}^{(k)}_{\text{transition}}(t)$: Regarding the movement policy of each particle $k \in \mathcal{K}$, it is basically composed of two distinct types: (i) a random movement term, modeled by the matrix $\mathbb{P}^{(k)}_{\text{rand}}$, which permits the particle to adventure through the network, without accounting for the defense of the previously dominated vertices; (ii) a biased movement term, modeled by the matrix $\mathbb{P}^{(k)}_{\text{bias}}$, 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 the stochastic vector $p(t) = [p^{(1)}(t), p^{(2)}(t), \dots, p^{(K)}(t)]$, which denotes the localization of the set of K particles presented to the network, where the kth-entry, $p^{(k)}(t)$, indicates the location of the particle k in the network at time t, i.e., $p^{(k)}(t) \in \mathcal{V}, \forall k \in \mathcal{K}$. It is desired to find a transition matrix that governs the probability distribution of the particles' movement to the immediate future state, $p(t+1) = [p^{(1)}(t+1), p^{(2)}(t+1), \dots, p^{(K)}(t+1)]$.

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.

With the intent of modeling such dynamics, we introduce the following random quantity $S^{(t)} = [S^{(1)}(t), \dots, S^{(K)}(t)]$, where the kth-entry, $S^{(k)}(t) \in \{0, 1\}$, indicates whether the particle k is dead or alive at time t. Specifically, if $S^{(k)}(t) = 1$, then particle k is said to be dead. Likewise, when $S^{(k)}(t) = 0$, the particle is said to be alive. Thus, if $S^{(k)}(t) = 0$, the particle navigates in the network according to a combined behavior of randomness and biased movement towards the dominated vertices. However, if $S^{(k)}(t) = 1$, the particle switches its movement policy to a new transition matrix, here entitled $\mathbb{P}^{(k)}_{\text{res}}(t)$, 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 k as:

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

where $\lambda \in [0, 1]$ indicates the desired fraction of biased movement that all particles in the network will perform, $\mathbb{P}^{(k)}_{\text{bias}}(t)$ portrays the transition matrix with a probability distribution according to the biased behavior described above and, likewise, $\mathbb{P}^{(k)}_{\text{rand}}(t)$ describes the random behavior, $S^{(k)}(t)$ indicates whether particle k is alive or dead, and $\mathbb{P}^{(k)}_{\text{res}}(t)$ is responsible for the particle resurrection behavior. Specifically, $\mathbb{P}^{(k)}_{\text{transition}}(i, j, t)$ indicates the probability that particle k makes a transition from vertex i to j at time t.

2) Determination of $\mathbb{P}_{\text{rand}}$: 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 $(i, j) \in \mathcal{V} \times \mathcal{V}$ of the matrix $\mathbb{P}^{(k)}_{\text{rand}}(t)$ is given by:

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

where $a_{i,j}$ denotes the (i, j)th-entry of the adjacency matrix A of the graph. Note that (2) resembles the traditional Markovian matrix for a single random walker, here symbolized as a particle [12]. Also note that matrix $\mathbb{P}^{(k)}_{\text{rand}}(t)$ is time-invariant and is the same for every particle in the network; therefore, we will drop the superscript k 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.

3) Determination of $\mathbb{P}^{(k)}_{\text{bias}}(t)$: In order to assist in the calculation of the matrix associated to the biased movement term, $\mathbb{P}^{(k)}_{\text{bias}}(t)$, for a given particle $k \in \mathcal{K}$, we introduce the following random quantity: $N_i(t) \triangleq [N_i^{(1)}(t), N_i^{(2)}(t), \dots, N_i^{(K)}(t)]$, where $\dim(N_i(t)) = 1 \times K$ and $N_i(t)$ stands for the number of visits received by vertex i up to time t (included) by all the particles scattered throughout the network. Specifically, the kth-entry, $N_i^{(k)}(t)$, indicates the number of visits made by the particle k 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: $N(t) \triangleq [N_1(t), N_2(t), \dots, N_V(t)]^T$ 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: $\bar{N}_i(t) \triangleq [\bar{N}_i^{(1)}(t), \bar{N}_i^{(2)}(t), \dots, \bar{N}_i^{(K)}(t)]$, 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 (included). Particularly, the kth-entry, $\bar{N}_i^{(k)}(t)$, indicates the relative frequency of visits performed by particle k 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: $\bar{N}(t) \triangleq [\bar{N}_1(t), \bar{N}_2(t), \dots, \bar{N}_V(t)]^T$, where $\dim(N(t)) = V \times K$. Mathematically, we define each entry of $\bar{N}_i^{(k)}(t)$ as:

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

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

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

Clearly, from (4), it can be observed that each particle has a different transition matrix associated to its biased 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.

4) Determination of $\mathbb{P}_{\text{res}}^{(k)}(t)$: Now we define each entry of $\mathbb{P}_{\text{res}}^{(k)}(t)$ that is accounted for teleporting a dead particle $k \in \mathcal{K}$ back to its owned territory. Suppose that particle k is visiting vertex i when its energy is completely depleted. In this situation, the particle teleports back to an arbitrary vertex j of its possession according to the probability given by:

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

where $\arg\max\limits_{m \in \mathbb{K}}(.)$ 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 (5) shows the probability of returning to an arbitrary vertex j of the particle's possession follows a uniform distribution. Moreover, all rows of this matrix are equal, showing that this movement does not depend on which vertex a specific particle is. This provides a compact way of computationally representing such structure. With that in mind, (5) only results in non-zero transition probabilities for vertices j that are being dominated by particle k, regardless of the existence of a connection between i and j in the adjacency matrix. In essence, once the particle is dead, the switch is enabled, which, in turn, compels the particle k to return to its previously owned territory, no matter if there is a physical connection or not in the adjacency matrix. If no vertex is being dominated by particle k at time t, we deliberately put it in any vertex of the network in a random manner, using a uniform distribution.

5) Particle's Energy Calculation and Update Rule: Now we proceed to the development of the particle's energy update rule. Firstly, it is useful to introduce the stochastic vector $E(t) = [E^{(1)}(t), \ldots, E^{(K)}(t)]$, where the kth-entry, $E^{(k)}(t) \in [\omega_{\min}, \omega_{\max}]$, $\omega_{\max} \geq \omega_{\min}$, denotes the energy level of particle k at time t, whose update rule is given by:

$$E^{(k)}(t) = \begin{cases} \min(\omega_{\max}, E^{(k)}(t-1) + \Delta), & \text{if owner}(k,t) \\ \max(\omega_{\min}, E^{(k)}(t-1) - \Delta), & \text{if not owner}(k,t) \end{cases} \tag{6}$$

where $\text{owner(k, t)} = \left(\arg\max\limits_{m \in \mathcal{K}}\left(\bar{N}_{p^{(k)}(t)}^{(m)}(t)\right) = k\right)$ is a logical expression that essentially yields true if the vertex that particle k visits at time t (i.e., vertex $p^{(k)}(t)$) is being dominated by 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 (6) represents the increment of the particle's energy and occurs when the particle k visits a vertex $p^{(k)}(t)$ which is dominated by itself, i.e., $\arg\max\limits_{m \in \mathbb{K}}\left(\bar{N}_{p^{(k)}(t)}^{(m)}(t)\right) = k$. Similarly, the second expression in (6) portrays the decrement of the particle's energy and occurs when particle k visits a vertex $p^{(k)}(t)$ 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 k. 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 k 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 $S^{(k)}(t)$ 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 kth-entry of $S(t)$ can be precisely written as:

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

where $\dim(S(t)) = 1 \times K$. Specifically, $S^{(k)}(t) = 1$ if $E^{(k)}(t) = \omega_{\min}$ and 0, otherwise. As there is an upper limit for the random variable $E^{(k)}(t)$, it is clear that if particle k 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, $\omega_{\max}$, is 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 the cluster detection rate of the dynamical system.

C. The Stochastic Particle Competition 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: $X(t) = [N(t) \quad p(t) \quad E(t) \quad S(t)]^T$ and the proposed competitive dynamical system as:

$$\phi : \begin{cases} N_i^{(k)}(t+1) & = N_i^{(k)}(t) + \mathbb{1}_{\{p^{(k)}(t+1)=i\}} \\ E^{(k)}(t+1) & = \begin{cases} \min(\omega_{\max}, E_i^{(k)}(t) + \Delta), \\ \qquad\qquad \text{if owner}(k,t) \\ \max(\omega_{\min}, E_i^{(k)}(t) - \Delta), \\ \qquad\qquad \text{if not owner}(k,t) \end{cases} \\ S^{(k)}(t+1) & = \mathbb{1}_{\{E^{(k)}(t+1)=\omega_{\min}\}} \end{cases} \tag{8}$$

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 $N_i^{(k)}(t) \in [1,\infty), (i,k) \in \mathcal{S}$, where $\mathcal{S}$ 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 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.

Note that system ϕ is nonlinear. This occurs on account of the indicator function, which is nonlinear. The first equation of system ϕ is responsible for updating the number of visits at vertex i by particle k 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 $(i,k) \in \mathcal{S}$ and the second and third expressions must be performed for every $k \in \mathcal{K}$ with the intention of one properly derive the full state $X(t)$ of the system ϕ. One can also see that system ϕ is clearly Markovian, since it only depends on the present state to derive the future state.

D. The Initial Conditions of the System

In order to run system ϕ, we need a set of initial conditions. Firstly, particles are randomly put in the network, i.e., the initial values of $p(0)$ are set randomly. Due to the competition nature, the particles will separate from each other even if they are put together at the beginning. Regarding the initial condition of the system ϕ, $X(0)$, it is valuable to stress that, with the purpose of (3) to be well-defined, all terms in $\sum_{k=1}^{K} N_i^{(k)}(t)$ cannot be zero simultaneously. In this way, we arbitrary set the initial value of $N_i^{(k)}$ to 1, $\forall (i,k) \in \mathcal{S}$, with no loss of fairness in the competition process. However, we also have to distinguish two types of vertices: (i) vertices from which the particles have generated at time $t = 0$; (ii) all other vertices. In view of that, we suggest the following initial condition to the matrix $N(0)$:

$$N_i^{(k)}(0) = \begin{cases} 2, & \text{if particle } j \text{ is generated at vertex } i \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

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 $k \in \mathcal{K}$ start out with the same energy level given by:

$$E^{(k)}(0) = \omega_{\min} + \left(\frac{\omega_{\max} - \omega_{\min}}{K}\right) \tag{10}$$

Lastly, the variable that accounts for indicating whether the particle k is dead or alive at the initial step, $S^{(k)}(0)$, $\forall k \in \mathcal{K}$, is given by:

$$S^{(k)}(0) = 0 \tag{11}$$

i.e., we deliberately set alive all particles in the network in the beginning of the process.

E. The Algorithm

Algorithm 1 summarizes all the steps to iterate the system ϕ. Essentially, the algorithm accepts the data set ($data$) and three user defined parameters: the number of particles (K), the desired fraction of preferential movement (λ) and a stopping factor (ϵ). The algorithm is not sensitive neither to λ nor to ϵ. Usually, good results can be obtained by selecting an arbitrary value between 0.6 to 0.8 for λ. ϵ can be set to an arbitrary small value. In all simulations in this paper, we simply use $\epsilon = 0.05M_{V\times K}$, where $M_{V\times K}$ is a matrix with dimensions $V \times K$ full of ones. Note that the stopping criterion can also be defined as a certain number of iterations. The only parameter that needs to be calibrated according to each input data set is the number of particles K, whose determination will be described later.

Algorithm 1 The Particle Competition Pseudo-Algorithm.

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

III. Application: Clustering of Handwritten Digits and Letters

In this section, we provide an application of data clustering for our proposed method for two real-world data sets. Specifically, in Subsect. III-A, we derive a dissimilarity measure that we will use in the network formation step; in Subsect. III-B, we supply some metainformation about the data sets, as well as the parameters to be used in the clustering task; in Subsect. III-C, we show a method for determining the optimal number of particles to be inserted into the network; and Subsect. III-D reveals the data clustering results that we have obtained from our algorithm applied to the MNIST and Letter Recognition databases. It is worth noting that in all simulations in this section, since the parameters $\omega_{\min}$, $\omega_{\max}$, and Δ are not sensitive to the model performance, we usually fix Δ as being 5% of the interval $\omega_{\max} - \omega_{\min}$. Therefore, we set $\Delta = 0.05$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$.

A. The Network Formation Technique

In this section, we describe the proposed dissimilarity measure, which is appropriate to be used in a graph-based data representation. In this situation, the images (data items) are portrayed by the vertices, while the data relationships are given by the links. A link holds a weight that numerically translates the similarity between the two vertices (images) at each end of it. Each image can be seen as "square" matrix $\eta \times \eta$, where η stands for the width and the height of the image. For rectangle images, a pre-processing is required to make its width and height equal. We conventionally set the range of any pixels to lie within the interval $[0, 1]$ by merely a normalization procedure. With that in mind, an arbitrary data item of the problem, say x_i, can be seen as a matrix with dimensions $\eta \times \eta$, where $x^{(u,j)} \in [0, 1], \forall (u, j) \in \{1, \ldots, \eta\} \times \{1, \ldots, \eta\}$.

In order to construct the network, we are required to establish a similarity measure. The traditional pixel-per-pixel distance is rather insufficient in terms of reliably representing data, since such measure is very sensitive to rotations and scale alterations. With the purpose of overcoming this difficulty, we propose a measure based on the eigenvalues that each image inherently carries with it. First of all, we remove the mean associated to each data item (image), so that we have a common basis of comparison. After that, we calculate the ϕ greatest eigenvalues of the image. The magnitudes of the eigenvalues are related to the variations that the image possesses, hence it is a natural carrier of information. The greater its value, more information about the image it conveys. In virtue of that, a good choice is to only extract the greatest $\phi < \eta$ eigenvalues and drop the smaller values, since these do not transport too much information about the image. In order to give more emphasis to the largest eigenvalues in an ordered manner, we associate weights to each eigenvalue position. In other words, during the comparison process, the difference between the first eigenvalues (the largest eigenvalues) of each image will have a higher weight than the difference of the second eigenvalues of each image, and so on, in a strictly decreasing order.

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

$$\rho(i, j) = \frac{1}{\rho_{max}} \sum_{k=1}^{\phi} \beta(k) \left[|\lambda_i^{(k)}| - |\lambda_j^{(k)}| \right]^2 \tag{12}$$

where $\rho \in [0, 1]$, $\rho_{max} > 0$ is a normalization constant, $\beta : \mathbb{N}^* \to (0, \infty)$ indicates a monotonically decreasing function that can be arbitrary chosen by the user. In another words, $\beta(1) > \beta(2) > \ldots > \beta(\phi)$, which mathematically reflects the importance of the ordering of the eigenvalues. If two images are similar, then it is expected that the difference between each pair of the first ϕ eigenvalues to be small, hence (12) will yield a small value. On the other hand, when we take into account two very distinct images i and j, their eigenvalues will be somewhat different, yielding a big value for $\rho(i, j)$.

As a final motivating remark, about elements with high dimensionality, the used similarity measure is of utter importance. In the case of this weighted eigenvalue similarity measure, it is independent on the dimensionality of the data. For instance, if we use a Minkowski measure, then as the dimensionality of the data increases, the quality of the algorithm will decrease, since the Minkowski measure is highly dependent on the data dimensionality.

B. Overview of the Data Sets

In this work, we have used 2 well-known data sets in order to show the effectiveness of our model, which are detailed as follows.

- *The modified Modified NIST data set* [13], which is composed of real handwritten numerical digits ("0" to "9"). Specifically, database originally comprises a training set with 60.000 samples and a test set of 10.000. Each sample is composed of an image of dimensions 28×28. In this work, the clustering task will be performed solely on the test set. Additionally, we have conducted a pre-processing step over the samples. In this case, we will normalize all the gray-level pixels from the image and reduce its size to fit in a 20×20 pixel box, while preserving their aspect ratio. In this simulation, we will make use of the dissimilarity measure based on the first 4 eigenvalues of each image out of 20 eigenvalues, since the image has dimensions 20×20. In virtue of that, we will employ as the function β in (12) an exponential decreasing function with a time constant fixed at $\tau = 3$ and a scaling factor given by 16, i.e., $\beta(x) = 16 \exp(\frac{x}{3})$. Since, this function is mapped into the interval $(0, \infty)$ and is a monotonically decreasing function, it follows that this β function meets all the aforementioned requirements. Specifically in this situation, we have that the weights associated to each eigenvalue are: $\beta(1) = 11.46$, $\beta(2) = 8.21$, $\beta(3) = 5.89$, and $\beta(4) = 4.22$. With the aid of such dissimilarity measure, we construct the graph using the k-nearest neighbor technique with $k = 3$.
- *The Letter Recognition data set* [14], which comprises 20.000 samples, each possessing a characteristic vector of 16 attributes. In this case, there are 26 classes to

be recognized, each one representing one letter of the alphabet (“A” to “Z”). The dissimilarity measure to be used will be the plain Euclidean distance. Regarding the network formation technique, we employ the k-nearest neighbor with $k = 6$.

C. Determining the Optimal Number of Particles and Clusters

In this section, we present a method for determining the optimal number of particles to be inserted into the network. Since the number of clusters is unknown a priori, we expect that the number of particles in the system to be exactly equal to the number of estimated particles. We reinforce that this is an intrinsic method that can be naturally captured from the competitive model itself. This happens on account of the rich set of information that the model implicitly and explicitly convey.

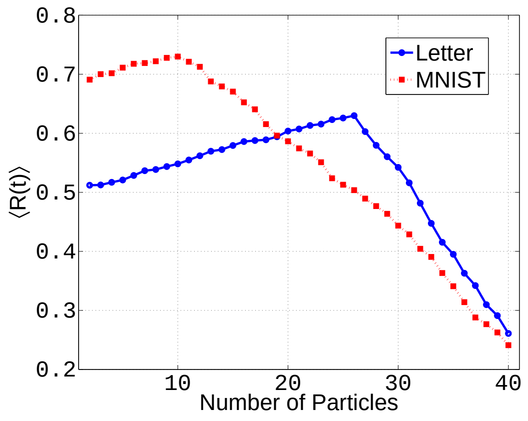
In light of that, we introduce a useful network measure denominated average maximum domination level $\langle R(t)\rangle$ as:

$$\langle R(t)\rangle = \frac{1}{V}\sum_{u=1}^{V}\max_{m\in\mathcal{K}}\left(\bar{N}_u^{(m)}(t)\right) \qquad (13)$$

where $\langle R(t)\rangle \in [0,1]$. When $\langle R(t)\rangle$ is near 1, we can infer that the competition in the network for vertices has ceased and, thus, the vertices' ownerships have been properly defined. In another words, the greatest domination level of any vertex possesses a number near 1, in average, showing that this vertex is being completely dominated by only one particle. This suggests that the clusters have been properly discovered and dominated. On the other hand, when $\langle R(t)\rangle$ approximates 0, we have that an intense competition is taking place into the network. In other words, the domination levels imposed by each particle in any arbitrary vertex is almost equal, in average. With these considerations in mind, a good suggestion for the optimal K in a given network is exactly when $\langle R(t)\rangle$ reaches its maximum value. Figure 1a shows the results obtained from the MNIST database. Indeed, the average maximum domination level $\langle R(t)\rangle$ reached its maximum value precisely when the number of particles is equivalent to the number of clusters in the problem, confirming our conjecture about the effectiveness of $\langle R(t)\rangle$ to establish the optimal number of particles in the model.

D. Clustering on the MNIST and Letter Recognition Databases

In this section, we provide the cluster detection accuracy reached by our algorithm in details, along with the detection accuracy of a selected set of competing techniques. For the calculation of this measure, we set that the ideal result is that each cluster represents a “digit” (in the MNIST database) or a “letter” (in the Letter Recognition database). Particularly, Table I supplies details about the algorithms that we have chosen for comparison matters. We have used the genetic algorithm available in the Global Optimization Toolbox of MATLAB with its default parameters with the goal of optimizing the parameters of the algorithms. In our case, we have optimized λ over the range $0.2 \le \lambda \le 0.8$. The Genetic Optimization Algorithm supplies the following optimal parameters: $\lambda = 0.58$. Table II reports the data clustering accuracy reached by each of these algorithms. Some of these results are readily extracted from [15] and [16]. For more information about the parameters, see the aforementioned references. In order to compare the algorithm, we use the mean rank. In order to calculate it, for each data set we rank the algorithms. In the end, for each algorithm, we take the mean of the rank achieved by each algorithm. As we can verify by looking at the mean rank results, our algorithm has reached one of the best positions, showing the effectiveness of the proposed technique.



(a)

Fig. 1. The methodology applied for determining the optimal number of particles (and the number of clusters) to be inserted into the model. We have used $\lambda = 0.6$ in this simulation. One can verify that the optimal number of particles coincides with the number of clusters in the network in both data sets, which is, in this case, 10 clusters for MNIST and 26 clusters for Letter Recognition. This confirms our prediction about the construction of the average maximum domination level $\langle R(t)\rangle$. We have run 5 independent simulations and have taken the average value for each point in the trace.

TABLE I
INFORMATION ABOUT A SELECTED SET OF DATA CLUSTERING TECHNIQUES.

Technique	**Reference**
Gaussian Mixture Model (GMM)	[17]
K-Means	[18]
Locally Consistent Gaussian Mixture Model (LCGMM)	[16]
Spectral clustering algorithm with normalized cut (Ncut)	[19]
Ncut Embedding All (NcutEmb$^{\text{All}}$)	[15]
Ncut Embedding Maximum (NcutEmb$^{\text{Max}}$)	[15]

Even though by Table II, it does not seem significant the results obtained by our algorithm, the proposed method has some advantages: (i) low computational complexity (for community detection tasks, the technique itself is $O(V)$ for sparse graphs and $O(V^2)$ for dense graphs; whereas, for data clustering, it is stacked at $O(V^2)$ - these points will be studied in an extended version), (ii) the information that the system carries is very rich, for example, we can derive measures to capture the number of clusters that an arbitrary data set has,

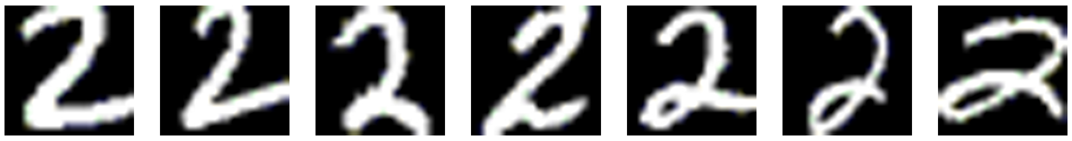


Fig. 2. A broad set of samples that are classified as being member of the cluster representing the pattern "2". Note that samples that are adjacent are similar with regards to the weighted eigenvalue dissimilarity function. The transitions from the sample (a) to (g) are captured from the maximum geodesic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

TABLE II
DATA CLUSTERING ACCURACY REACHED BY A SELECTED SET OF ALGORITHMS. 10 INDEPENDENT RUNS ARE PERFORMED FOR EACH STOCHASTIC-BASED TECHNIQUE AND THE CORRESPONDING MEAN IS PROVIDED.

	Letter Recognition	**MNIST**	**Mean Rank**
LCGMM	73.60	93.03	**2.5**
GMM	66.60	91.24	**4.5**
K-Means	53.10	87.94	**7.0**
NCut	68.80	88.72	**5.5**
NCutEmb$^{\text{All}}$	75.10	90.07	**3.5**
NCutEmb$^{\text{Max}}$	75.63	90.59	**2.5**
Proposed Technique	74.53	91.37	**2.5**

or we can even find overlapping vertices in the graph, among many other possibilities.

In order to further verify the robustness of the proposed technique, we inspect the members of the cluster representing the pattern "2". Figure 2 depicts some examples that have been extracted from this cluster. These samples are captured using the following strategy: we compute the vertices that compose the maximum geodesic distance of the cluster representing pattern "2" (diameter), which, in this case, is 17. We only show 7 representative samples. Samples that are adjacent are more similar than those distant one from another. On the basis of this analysis, we conclude that the graph representation has successfully captured several variations of the pattern "2", showing the robustness of the proposed model.

IV. Conclusions and Future Work

This paper proposes a 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 navigate in the network to explore their territory and, at the same time, attempt to defend its territory against rival particles. Additionally, we have derived an embedded technique for determining the right number of clusters in a data set, which happens to be exactly the number of particles to be inserted into the network. As this is evaluated from the competition process itself, no extra processing is necessary. Finally, simulations have been carried out on handwritten digits and letters recognition and promising results have been obtained, showing that the proposed competitive model is effective for data clustering tasks.

As future work, we are aiming at proposing some applications and extensions of the model, such as (i) detection of overlapping structures or vertices in the network, (ii) usage of different number of particles to provide hierarchical clustering, (iii) extension of the algorithm for the semi-supervised approach, among others.

Acknowledgment

This work is supported by the So Paulo State Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq).

References

[1] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
[2] B. Kosko, "Stochastic competitive learning," *IEEE Trans. Neural Networks*, vol. 2, no. 5, pp. 522–529, 1991.
[3] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, pp. 23–63, 1987.
[4] L. C. Jain, B. Lazzerini, and U. H. (eds.), *Innovations in ART Neural Networks (Studies in Fuzziness and Soft Computing)*. Physica-Verlag, Heidelberg, 2010.
[5] G. Deboeck and T. K. (eds.), *Visual Explorations in Finance: with Self-Organizing Maps*. Springer, 2010.
[6] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.
[7] 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.
[8] M. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.
[9] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.
[10] M. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, p. 036104, 2006.
[11] ——, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, p. 066133, 2004.
[12] E. Çinlar, *Introduction to Stochastic Processes*. Englewood Cliffs, N. J.: Prentice-Hall, 1975.
[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[14] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.
[15] F. Ratle, J. Weston, and M. L. Miller, "Large-scale clustering through functional embedding," in *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ser. ECML PKDD '08. Springer-Verlag, 2008, pp. 266–281.
[16] J. Liu, D. Cai, and X. He, "Gaussian mixture model with local consistency," in *AAAI'10*, vol. 1, 2010, pp. 512–517.
[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[18] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, 1967, pp. 281–297.
[19] J. Shi and J. Malik, "Normalized cut and image segmentation," Berkeley, CA, USA, Tech. Rep., 1997.