

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/224946854>

Handwritten Data Clustering Using Agents Competition in Networks

Article in *Journal of Mathematical Imaging and Vision* · October 2012

DOI: 10.1007/s10851-012-0353-z

CITATIONS

15

READS

104

3 authors:



Thiago Christiano Silva

Banco Central do Brasil

156 PUBLICATIONS 2,196 CITATIONS

[SEE PROFILE](#)



Liang Zhao

University of São Paulo

406 PUBLICATIONS 8,344 CITATIONS

[SEE PROFILE](#)



Thiago Henrique Cupertino

Secretary of Treasure

26 PUBLICATIONS 193 CITATIONS

[SEE PROFILE](#)

Handwritten Data Clustering Using Agents Competition in Networks

Thiago C Silva · Liang Zhao · Thiago H. Cupertino

Published online: 12 June 2012
© Springer Science+Business Media, LLC 2012

Abstract In this paper, we study a new type of competitive learning scheme realized on large-scale networks. The model consists of several agents walking within the network and competing with each other to occupy as many nodes as possible, while attempting to reject intruder agents. In the end of the process, each agent dominates a community (a strongly connected subnetwork). Here, the model is described by a stochastic dynamical system. In this paper, a mathematical analysis for uncovering the system's properties is presented. In addition, the model is applied to solve handwritten digits and letters clustering problems. An interesting feature is that the model is able to group the same digits or letters even with considerable distortions into the same cluster. Computer simulations reveal that the proposed technique presents high precision of cluster detections, as well as low computational complexity.

Keywords Stochastic competitive learning · Clustering · Handwritten recognition · Interacting agents

1 Introduction

Data clustering is an unsupervised learning task that aims at discovering a natural grouping of a set of patterns, points,

or objects by means of a similarity measure [11, 16]. Each cluster is a collection of data items which are similar between themselves and are dissimilar to the objects belonging to other clusters. Data clustering is vital in several exploratory pattern-analysis, decision-making, and machine-learning situations. Some of them include data mining, document retrieval, image segmentation, bioinformatics, and pattern classification [5, 8, 15–17]. Unfortunately, in the majority of such tasks, little prior information is available about the data. In this way, advances in the methodology to automatically understand, process, and summarize the data are required. Nowadays, this becomes even more critical by virtue of the exponential increase in both the volume and the variety of data [16]. In this scenario, the decision-maker must perform as few assumptions about the data as possible. It is under these practical restrictions that the clustering procedure is especially appropriate for the exploration of inter-relationships among the data points to make an assessment (perhaps preliminary) of their structure [16, 17]. Data clustering algorithms are generally divided in two types: hierarchical or partitional [2, 8, 16]. The former finds successive clusters using previously established clusters, whereas the latter determines all clusters at once. Hierarchical algorithms can be agglomerative (“bottom-up”) or divisive (“top-down”). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Two-way clustering, co-clustering or bi-clustering are the names for clusterings where not only the objects are clustered but also the features of the objects, i.e., if the data is represented in a data matrix, the row and columns are clustered simultaneously [11].

One of the recent approaches of data clustering is based on graphs (networks). In this context, a network is firstly

T. C Silva (✉) · L. Zhao · T.H. Cupertino
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@icmc.usp.br

L. Zhao
e-mail: zhao@icmc.usp.br

T.H. Cupertino
e-mail: thiagohc@icmc.usp.br

constructed from the original data set; then, the network is partitioned into modular subnetworks, also called communities, producing data clusters. The notion of community in networks is straightforward: each community is defined as a sub-graph whose nodes are densely connected within itself, but sparsely connected with the rest of the network. Community detection in networks has turned out to be an important topic in graph mining and data mining [6, 9, 25]. 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 has been spent to develop efficient but suboptimal solutions, such as the spectral methods [26], the technique based on the “betweenness” measure [25], modularity optimization [24], community detection based on the Potts model [30], synchronization [1], information theory [10], and random walks [36]. For a recent review of this topic, see [9]. The main motivation of graph theory research is the ability to describe the topological structure of the original system. In the machine learning domain, it has been shown that the topological structure is quite useful to detect clusters of arbitrary forms in data clustering [9, 19], in contrast to the widely used data clustering technique—K-Means, where only spherical-shaped clusters can be identified.

Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touchscreens, data sets, and other devices [20, 34]. Ideally, the handwriting recognition systems should be able to read and understand any handwriting [3]. Handwriting recognition has been one of the most fascinating and challenging research areas in the field of image processing and pattern recognition in the recent years [27]. It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications [3, 34]. In general, handwritten recognition is classified into off-line or on-line. In the first case, the writing is obtained by an electronic device and the captured writing is completely available as an image to the handwritten recognition method. In the second case, the coordinates of successive points are available by means of a function dependent on time, i.e., the complete image is not given [27, 34]. In summary, several research works have been proposed [13, 23, 27] in an attempt to reduce the processing time of both off-line and on-line methods, while, at the same time, providing higher recognition accuracy. Due to the high complexity that this topic offers, it still has a wide range of problems to be addressed, such as the efficient recognition of images that are distorted or suffered a non-linear transformation [3, 8, 34]. Pattern clustering is a fundamental step in a pattern recognition system. The recognition performance is strongly dependent on the quality of pattern grouping and labeling.

In this paper, we study handwritten data clustering by using a new type of competitive learning mechanism [28, 32, 33]. Consider a large-scale graph (network), where several agents walk in the network and compete with each other to occupy as many nodes as possible, while attempting to reject intruder agents. Each agent can perform a random walk by choosing any neighbor to visit, a preferential walk by choosing the node with the highest domination to visit, or a combination of both. Straightforward applications are community detection and data clustering. The competitive walking process reaches a dynamic equilibrium when each community or data cluster is dominated by a single agent.

The agent competition model was originally proposed in [28, 32, 33], where only a procedure of agent competition is introduced and no mathematical analysis is presented. In the present work, we study the agent competition model represented by a nonlinear stochastic dynamical system. Moreover, a mathematical analysis for uncovering the properties of the dynamical system is provided. In particular, we show that the competitive dynamical system reduces to multiple independent random walks when the parameters, which characterize the competition and reanimation features, are turned off. Since the models of several interactive walking processes correspond to many natural and artificial systems, and due to the lack of theory for such models, our approach is an important step to understand such systems. It is worth mentioning that the underlying network is constructed directly from the input data set; therefore, the correspondence between the input data and the processing result (the final network) is maintained. Besides this, we apply the competitive learning model to solve handwritten data clustering problems. An interesting feature is that the model is able to group the same digits or letters even with considerable distortions into the same cluster.

The remainder of the paper is organized as follows. The competitive model is reviewed in Sect. 2. In Sect. 3, a mathematical analysis is provided. Section 4 presents an application of the competitive model in handwritten data clustering. Finally, Sect. 5 concludes the paper.

2 Model Description

In this section, we review the competitive learning technique.

2.1 The Competitive Transition Matrix

Consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \{v_1, \dots, v_V\}$ is the set of vertices (or nodes) and $\mathcal{E} = \{e_1, \dots, e_L\} \subset \mathcal{V} \times \mathcal{V}$ is the set of links (or edges). In the proposed competitive learning model, a set of agents $\mathcal{K} = \{1, \dots, K\}$ is inserted into the vertices of the network in a random manner. Each agent

can be conceptualized as a flag carrier with its main objective being to conquer new vertices, while defending its current dominated vertices. In this case, a competition process will naturally take place amongst the agents. When an agent visits an arbitrary vertex, it strengthens its own domination level on that vertex and, simultaneously, weakens the domination levels of all other rival agents on the same vertex. At the end, each agent dominates a community.

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

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

$$\begin{aligned} \mathbb{P}_{\text{transition}}^{(k)}(t) &\triangleq (1 - S^{(k)}(t)) [\lambda \mathbb{P}_{\text{pref}}^{(k)}(t) + (1 - \lambda) \mathbb{P}_{\text{rand}}^{(k)}] \\ &\quad + S^{(k)}(t) \mathbb{P}_{\text{rean}}^{(k)}(t), \end{aligned} \quad (1)$$

where $\lambda \in [0, 1]$ indicates the desired fraction of preferential movement that all agents in the network will perform. Specifically, $\mathbb{P}_{\text{transition}}^{(k)}(i, j, t)$ indicates the probability that agent k performs a transition from vertex i to j at time t . We now define each matrix that appears in (1) in a detailed manner.

The random movement matrix only depends on the adjacency matrix of the graph. In this way, each entry $(i, j) \in$

$\mathcal{V} \times \mathcal{V}$ of the matrix $\mathbb{P}_{\text{rand}}^{(k)}$ is given by:

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

where $a_{i,j}$ denotes the (i, j) th-entry of the adjacency matrix A of the graph. Note that matrix $\mathbb{P}_{\text{rand}}^{(k)}$ is time invariant and it is the same for every agent in the network; therefore, we drop the superscript k . In short, the probability of an adjacent neighbor j to be visited from vertex i is proportional to the edge weight linking these two vertices.

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

$$N_i(t) \triangleq [N_i^{(1)}(t), N_i^{(2)}(t), \dots, N_i^{(K)}(t)]^T, \quad (3)$$

where $\dim(N_i(t)) = K \times 1$, T denotes the transpose operator, and $N_i(t)$ stands for the number of visits received by vertex i up to time t by all agents scattered throughout the network. Then, the matrix that contains the number of visits made by every agent in the network to all the vertices is defined as:

$$N(t) \triangleq [N_1(t), N_2(t), \dots, N_V(t)]^T, \quad (4)$$

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

$$\bar{N}_i(t) \triangleq [\bar{N}_i^{(1)}(t), \bar{N}_i^{(2)}(t), \dots, \bar{N}_i^{(K)}(t)]^T, \quad (5)$$

where $\dim(\bar{N}_i(t)) = K \times 1$ and $\bar{N}_i(t)$ denotes the relative frequency of visits of all agents in the network to vertex i at time t . Similarly to the previous case, this notation is extended to all vertices in the network by defining the domination level matrix that sustains all the domination levels imposed by every agent in the network to all the vertices as:

$$\bar{N}(t) \triangleq [\bar{N}_1(t), \bar{N}_2(t), \dots, \bar{N}_V(t)]^T, \quad (6)$$

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

$$\bar{N}_i^{(k)}(t) \triangleq \frac{N_i^{(k)}(t)}{\sum_{u=1}^K N_i^{(u)}(t)}. \quad (7)$$

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

$$\mathbb{P}_{\text{pref}}^{(k)}(i, j, t) \triangleq \frac{a_{i,j} \bar{N}_j^{(k)}(t)}{\sum_{u=1}^V a_{i,u} \bar{N}_u^{(k)}(t)}. \quad (8)$$

From (8), it can be observed that each agent has a different transition matrix associated to its preferential movement and that, unlike the matrix related to the random movement, it is time-variant with dependence on the domination levels of all the vertices ($\bar{N}(t)$) in the network at time t . It is worth mentioning that the approach taken here to characterize the preferential movement of the agents is defined as the visiting frequency of each agent to a specific vertex, meaning as more visits are performed by a specific agent to an arbitrary vertex, there will be a higher chance for the same agent to repeatedly visit the same vertex. The consequence of this behavior is two-fold: (i) the visiting agent strengthens its the domination level on a vertex and (ii) weakens the domination levels of all other agents on the same vertex. This is made possible because of the normalization process that is conservative.

Now we define each entry of $\mathbb{P}_{\text{rean}}^{(k)}(t)$. Suppose that agent k is visiting vertex i when its energy is completely depleted. In this situation, the agent must regress to an arbitrary vertex j of its possession at time t , according to the following expression:

$$\mathbb{P}_{\text{rean}}^{(k)}(i, j, t) \triangleq \frac{\mathbb{1}\left\{ \arg \max_{m \in \mathcal{K}} (\bar{N}_j^{(m)}(t)) = k \right\}}{\sum_{u=1}^V \mathbb{1}\left\{ \arg \max_{m \in \mathcal{K}} (\bar{N}_u^{(m)}(t)) = k \right\}}, \quad (9)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function that yields 1 if the argument is logically true and 0, otherwise. The operator $\arg \max_{m \in \mathbb{K}}(\cdot)$ returns an index M , where $\bar{N}_u^{(M)}(t)$ is the maximal value among all $\bar{N}_u^{(m)}(t)$ for $m \in \mathcal{K}$. A careful analysis of the expression in (9) reveals that the probability of returning to an arbitrary vertex j that is dominated by the agent k follows a uniform distribution. In other words, (9) only results in non-zero transition probabilities for vertices j that are being dominated by agent k at time t , regardless of the existence of a connection between i and j in the adjacency matrix.

Once defined the collection of matrices associated to each agent, we couple all of them into a single representative transition matrix denominated $\mathbb{P}_{\text{transition}}(t)$ which models the transition of $p(t)$ to $p(t+1)$. Given $X(t)$, one can see that $p^{(k)}(t+1)$ and $p^{(u)}(t+1)$ are independent for every pair $(k, u) \in \mathcal{K} \times \mathcal{K}, k \neq u$. In this way, $\mathbb{P}_{\text{transition}}(t)$ can be written as:

$$\mathbb{P}_{\text{transition}}(t) = \mathbb{P}_{\text{transition}}^{(1)}(t) \otimes \cdots \otimes \mathbb{P}_{\text{transition}}^{(K)}(t), \quad (10)$$

where \otimes denotes the Kronecker tensor product operator. In this way, (10) completely specifies the transition distribution matrix for all the agents in the network.

Essentially, when $K \geq 2$, $p(t)$ will be a vector and we would no longer be able to conventionally define the row $p(t)$ of matrix $\mathbb{P}_{\text{transition}}(t)$. Owing to this, we define an invertible mapping $f: \mathcal{V}^K \mapsto \mathbb{N}$. The function f simply maps

the input vector to a scalar number that reflects the natural ordering of the tuples in the input vector. For example, $p(t) = [1, 1, \dots, 1, 1]$ (all agents at vertex 1) denotes the first state; $p(t) = [1, 1, \dots, 1, 2]$ (all agents at vertex 1, except the last agent, which is at vertex 2) is the second state; and so on, up to the scalar state V^K . Therefore, with this tool, we can fully manipulate the matrix $\mathbb{P}_{\text{transition}}(t)$.

2.2 The Unsupervised Competitive Learning Model

In light of the results obtained in the previous section, the internal state of the dynamical system, $X(t)$, is given by:

$$X(t) = \begin{bmatrix} p(t) \\ N(t) \\ E(t) \\ S(t) \end{bmatrix}, \quad (11)$$

and the proposed competitive dynamical system is given by:

$$\phi : \begin{cases} N_i^{(j)}(t+1) = N_i^{(j)}(t) + \mathbb{1}_{\{p^{(j)}(t+1)=i\}}, \\ E^{(j)}(t+1) = \begin{cases} \min(\omega_{\max}, E^{(j)}(t) + \Delta), & \text{if owner}(j, t), \\ \max(\omega_{\min}, E^{(j)}(t) - \Delta), & \text{if } \sim \text{owner}(j, t), \end{cases} \\ S^{(j)}(t+1) = \mathbb{1}_{\{E^{(j)}(t+1)=\omega_{\min}\}}. \end{cases} \quad (12)$$

An important characteristic of system ϕ , which will be extensively used later, is its Markovian property (see Proposition 1). In the following, we present the meaning of the 3 expressions shown in the competitive dynamical system ϕ :

- *Update Rule of the Number of Visits (1st Expression):* In this update rule, for every agent $k \in \mathcal{K}$ in the model, the vertex i that is being visited by agent k has its number of visits associated to that agent, $N_i^{(k)}(t)$, added by 1. If agent k does not visit vertex i at time t , then its number of visits remains unaltered. This two-fold behavior is achieved by the indicator function in the right-hand side of the first expression in ϕ . Specifically, when agent k visits vertex i , the argument in the indicator function returns true, and the expression becomes $N_i^{(k)}(t+1) = N_i^{(k)}(t) + 1$. Conversely, if agent k does not visit vertex i , then the argument in the indicator function yields false and the update rule of the number of visits reduces to $N_i^{(k)}(t+1) = N_i^{(k)}(t)$.

- *Update Rule of the Agent's Energy (2nd Expression):* Here, we introduce the stochastic vector: $E(t) = [E^{(1)}(t), \dots, E^{(K)}(t)]$, where the k th-entry, $E^{(k)}(t) \in [\omega_{\min}, \omega_{\max}]$, $\omega_{\max} \geq \omega_{\min}$, denotes the energy level of agent k at time t . In the second expression of system ϕ , $\text{owner}(k, t) = (\arg \max_{m \in \mathcal{K}} (\bar{N}_{p^{(k)}(t)}^{(m)}(t)) = k)$ is a logical expression that essentially yields true if the vertex that agent k visits at

time t (i.e., vertex $p^{(k)}(t)$) is being dominated by it, but yields false otherwise. In the same expression, $\Delta > 0$ symbolizes the increment or decrement of energy that each agent receives at time t . The first case represents the increment of the agent's energy and it occurs when agent k visits a vertex $p^{(k)}(t)$ which is dominated by itself, i.e., $\arg \max_{m \in \mathbb{K}} (\bar{N}_{p^{(k)}(t)}^{(m)}(t)) = k$. Similarly, the second case indicates the decrement of the agent's energy that happens when it visits a vertex dominated by rival agents. Therefore, in this model, agents will be given a penalty if they are wandering in rival territory, so as to minimize aimless navigation of the agents in the network.

- *Update Rule of the Agent's State (3rd Expression):* Here, the update rule that governs $S(t)$ is given. As we have stated, an arbitrary agent k will be transported back to its domain only if its energy drops to a threshold ω_{\min} . With that in mind, it is natural that each entry of $S^{(k)}(t)$ must monitor the current energy value of its corresponding agent k . If this energy ever drops to the given threshold, the switch must be enabled. Analogously, if the agent still has an energy value greater than this threshold, then the switch should be disabled. This behavior is exactly expressed in the third expression of the competitive system ϕ .

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. Specifically, the random numbers are generated respecting the probability distribution given in (1).

Finally, we will define the competitive system in a matrix form. This will be specially useful in the theoretical derivations that will be performed in the next section. Specifically, system ϕ can be written as:

$$\phi' : \begin{cases} N(t+1) = f_N(N(t), p(t+1)) \\ E(t+1) = f_E(N(t+1), p(t+1)) \\ S(t+1) = f_S(E(t+1)), \end{cases} \quad (13)$$

where $f_N(\cdot)$, $f_E(\cdot)$, and $f_S(\cdot)$ are suitable stochastic matrix functions as defined term by term in (12).

2.3 Computational Complexity

The competitive model reviewed in Sect. 2 has been shown to run in $\mathcal{O}(KV)$ time for community detection tasks (sparse graph) and in $\mathcal{O}(V^2)$ time for data clustering tasks (cf. [33]). It is worth noting that, based on [33, Table 1], we can visualize that the community detection technique has the lowest computational complexity order compared to all other techniques developed so far. With respect to data clustering tasks, if we look at [33, Table II], it has a reasonable complexity order (it is not the lowest one), mainly by virtue of the network formation step, which is time-demanding basically because of the distance matrix calculation method.

3 Theoretical Results

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

3.1 Mathematical Analysis

As the initial step, the transition probability function of system ϕ , i.e., $P(X(t+1) | X(t))$, is going to be derived. Note that the marginal probability of the system's state $P(X(t))$ can be written in terms of a joint probability of each of the components of the system's state, meaning $P(X(t)) = P(N(t), p(t), E(t), S(t))$. Using this fact and applying the Product Rule, one has:

$$P(X(t+1) | X(t)) = P_{p(t+1)} P_{N(t+1)} P_{E(t+1)} P_{S(t+1)}, \quad (14)$$

where:

$$P_{S(t+1)} = P(S(t+1) | N(t+1), p(t+1), E(t+1), X(t)), \quad (15)$$

$$P_{E(t+1)} = P(E(t+1) | N(t+1), p(t+1), X(t)), \quad (16)$$

$$P_{N(t+1)} = P(N(t+1) | p(t+1), X(t)), \quad (17)$$

$$P_{p(t+1)} = P(p(t+1) | X(t)). \quad (18)$$

Next, the algebraic derivation of these four quantities is given in a detailed manner.

3.1.1 Obtaining $P_{p(t+1)}$

Noting that the stochastic vector $p(t+1)$ is directly evaluated from $\mathbb{P}_{\text{transition}}(t)$ given in (10), which in turn only requires $p(t)$ and $N(t)$ to be constructed ($X(t)$ is given), then the following equivalence holds:

$$P_{p(t+1)} = P(p(t+1) | X(t)) = \mathbb{P}_{\text{transition}}(N(t), p(t)). \quad (19)$$

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

3.1.2 Obtaining $P_{N(t+1)}$

In this case, the system's state $X(t)$ and $p(t+1)$ are given. Observing that the determination of $N(t+1)$ only depends on $N(t)$ and $p(t+1)$ (see first expression in system ϕ), it follows that $P(N(t+1))$ can be completely evaluated. In view of this, the following expression holds:

$$\begin{aligned} P_{N(t+1)} &= P(N(t+1) | p(t+1), X(t)) \\ &= \mathbb{1}_{\{N(t+1)=N(t)+Q_N(p(t+1))\}}, \end{aligned} \quad (20)$$

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

$$Q_N(p(t+1))(i, k) = \mathbb{1}_{\{p^{(k)}(t+1)=i\}}. \quad (21)$$

The argument in the indicator function shown in (20) is essentially the first expression of system ϕ , but with a matrix notation. In brief, (20) will result in 1 if the computation of $N(t+1)$ is correct, given $p(t+1)$ and $N(t)$, i.e., it is in compliance with the dynamical system rules; and 0, otherwise.

3.1.3 Obtaining $P_{E(t+1)}$

Here, the system's state $X(t)$, $p(t+1)$, and $N(t+1)$ are given. By inspecting (7), one can see that the calculation of $\bar{N}(t)$ only depends on $N(t)$. As a result, one can consider $\bar{N}(t)$ as given variable too. Due to this fact, together with the second expression in ϕ , one can see that $E(t+1)$ can be completely determined since $E(t)$, $p(t+1)$, and $\bar{N}(t+1)$ are known quantities. On account of that, $P_{E(t+1)}$ is given by:

$$\begin{aligned} P_{E(t+1)} &= P(E(t+1) | N(t+1), p(t+1), X(t)) \\ &= \mathbb{1}_{\{E(t+1)=E(t)+\Delta \times Q_E(p(t+1), N(t+1))\}}, \end{aligned} \quad (22)$$

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

$$\begin{aligned} Q_E^{(k)}(p(t+1), N(t+1)) \\ = \mathbb{1}_{\{\text{owner}(k, t+1)\}} - \mathbb{1}_{\{\sim \text{owner}(k, t+1)\}}. \end{aligned} \quad (23)$$

Note that the argument of the indicator function in (23) is essentially the second expression of ϕ in a compact matrix form. Indicator functions were employed to describe the two types of behaviors that this variable can have: an increment or decrement of the agent's energy. Suppose that agent $k \in \mathcal{K}$ is visiting a vertex that is being owned by itself, then only the first indicator function in (23) will be enabled; hence, $Q_E^{(k)}(p(t+1), N(t+1)) = 1$. Similarly, if agent k is visiting a vertex that is being owned by a rival agent, then the second indicator function is enabled, yielding $Q_E^{(k)}(p(t+1), N(t+1)) = -1$. This behavior together with (22) is exactly the second expression of ϕ in a compact matrix form.

3.1.4 Obtaining $P_{S(t+1)}$

Now, the system's state $X(t)$, $E(t+1)$, $N(t+1)$, and $p(t+1)$ are given. By analyzing the third expression in ϕ , one can verify that $S(t+1)$ can be determined once $E(t+1)$ is known. In this way, one can surely evaluate $P_{S(t+1)}$ in this scenario as follows:

$$\begin{aligned} P_{S(t+1)} &= P(S(t+1) | E(t+1), N(t+1), p(t+1), X(t)) \\ &= \mathbb{1}_{\{S(t+1)=Q_S(E(t+1))\}}, \end{aligned} \quad (24)$$

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

$$Q_S^{(k)}(E(t+1)) = \mathbb{1}_{\{E^{(k)}(t+1)=\omega_{\min}\}}. \quad (25)$$

3.1.5 The Transition Probability Function

Substituting (19), (20), (22), and (24) into (14), the transition probability function of the competitive dynamical system is given as follows:

$$\begin{aligned} P(X(t+1) | X(t)) &= \mathbb{1}_{\{N(t+1)=N(t)+Q_N(p(t+1))\}} \\ &\quad \times \mathbb{1}_{\{S(t+1)=Q_S(E(t+1))\}} \\ &\quad \times \mathbb{1}_{\{E(t+1)=E(t)+\Delta Q_E(p(t+1), N(t+1))\}} \\ &\quad \times \mathbb{P}_{\text{transition}}(N(t), p(t)) \\ &= \mathbb{1}_{\{\text{Compliance}(t)\}} \mathbb{P}_{\text{transition}}(N(t), p(t)), \end{aligned} \quad (26)$$

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

$\text{Compliance}(t)$

$$\begin{aligned} &= [N(t+1) = N(t) + Q_N(p(t+1))] \\ &\quad \wedge [S(t+1) = Q_S(E(t+1))] \\ &\quad \wedge [E(t+1) = E(t) + \Delta Q_E(p(t+1), N(t+1))], \end{aligned} \quad (27)$$

i.e., $\text{Compliance}(t)$ encompasses all the rules that have to be satisfied in order to the indicator functions in (26) produce 1. It is worth mentioning that non-zero transition probabilities will only happen when these rules are satisfied.

3.1.6 Obtaining $P(N(t))$

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

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

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

state trajectory, is denoted by:

$$\begin{aligned} P(X(t+1) \in X_{t+1} | X(t), \dots, X(0)) \\ = P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} \right. \\ \left. | X(t), \dots, X(0)\right). \end{aligned} \quad (28)$$

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

$$\begin{aligned} P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} | X(t), \dots, X(0)\right) \\ = P\left(p_{t+1} : \begin{bmatrix} f_N(N(t), p_{t+1}) \\ f_E(N(t+1), p_{t+1}) \\ f_S(E(t+1)) \end{bmatrix} \in X_{t+1} | X(t)\right) \\ = P(X(t+1) \in X_{t+1} | X(t)). \end{aligned} \quad (29)$$

Therefore, in view of (29), $\{X(t) : t \geq 0\}$ is a Markovian process, since it only depends on the present state to specify the next state and, hence, the past history of the system's trajectory is irrelevant. \square

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

$$\begin{aligned} P(X(0), \dots, X(t)) &= P(X(t) | X(t-1)) \\ &\quad \times P(X(t-1) | X(t-2)) \\ &\quad \times \dots \times P(X(1) | X(0))P(X(0)). \end{aligned} \quad (30)$$

Using the transition function that governs system ϕ , as illustrated in (26), to each shifted term in (30), we get:

$$\begin{aligned} P(X(0), \dots, X(t)) &= P(X(0)) \prod_{u=1}^{t-1} [\mathbb{1}_{\{\text{Compliance}(u)\}} \\ &\quad \times \mathbb{P}_{\text{transition}}(N(u), p(u))], \end{aligned} \quad (31)$$

where $P(X(0)) = P(N(0), p(0), E(0), S(0))$. Observing now that $P(X(0), \dots, X(t)) = P(N(0), p(0), E(0), S(0), \dots, N(t), p(t), E(t), S(t))$, we marginalize this joint distribution with respect to $N(t)$ as follows:

$$P(N(t)) = \sum_{\sim N(t)} P(X(0), \dots, X(t)) \quad (32)$$

where $\sim N(t)$ means that we sum over all the possible values of $X(0), \dots, X(t)$, except for $N(t)$ which is inside $X(t) = [N(t) \ p(t) \ E(t) \ S(t)]^T$. Using (31) in (32), we are able to obtain $P(N(t))$ as follows:

$$\begin{aligned} P(N(t)) &= \sum_{\sim N(t)} \left\{ P(X(0)) \prod_{u=1}^{t-1} [\mathbb{1}_{\{\text{Compliance}(u)\}} \right. \\ &\quad \left. \times \mathbb{P}_{\text{trans}}(N(u), p(u))]\right\}. \end{aligned} \quad (33)$$

3.1.7 Obtaining $P(\bar{N}(t))$

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

Example 1 Consider a network with 3 agents and 2 vertices. At time t , suppose that the stochastic process is able to produce two distinct configurations for $N(t)$, as follows:

$$\begin{aligned} N(t) &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}, \\ N'(t) &= \begin{bmatrix} 2 & 2 & 2 \\ 2 & 4 & 6 \end{bmatrix}. \end{aligned} \quad (34)$$

Then, the setups in (34) applied to (7) make clear that both configurations yield the same $\bar{N}(t)$ given by:

$$\bar{N}(t) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{3} & \frac{1}{2} \end{bmatrix}. \quad (35)$$

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

Following the aforementioned strategy, $P(\bar{N}(t))$ can be calculated by summing over all multiples of $uN(t)$, $u \in \{1, \dots, t\}$ such that $f(uN(t)) = \bar{N}(t)$, where f is the normalization function defined in (7). On account of this, we have:

$$P(\bar{N}(t) = U : U \in \mathcal{V} \times \mathcal{K}) = \sum_{u=1}^t P(f(uN(t)) = U). \quad (36)$$

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

3.2 The Proposed Competitive System Generalizes Multiple Independent Random Walks

In this section, we will demonstrate the following interesting property: when $\lambda = 0$ and $\Delta = 0$, the competitive mechanism among the agents and the reanimation feature are turned off, and the system reduces to multiple independent random walkers [4]. Conversely, when $\lambda > 0$, the competitive mechanism among the agents is enabled and the combination of random-preferential interacting walks occurs. In this case, the reanimation feature is presented depending on the choice of Δ .

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

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

$$p(t+1) = p(t)\mathbb{P}_{\text{transition}} \quad (37)$$

where $\mathbb{P}_{\text{transition}} = \mathbb{P}_{\text{rand}} \otimes \mathbb{P}_{\text{rand}} \otimes \cdots \otimes \mathbb{P}_{\text{rand}}$ and $p(t)$ is an enumerated state encompassing all the agents, as described before. Here, the independence among the agents is demonstrated by showing that the generated $N(t)$ by system ϕ is exactly the same as the one produced by the potential matrix of the Markov Chains Theory [4]. In other words, $N(t)$ can be implicitly calculated from the stochastic process $\{p(t) : t \geq 0\}$.

First, it is useful to find a closed expression for $N(t)$ in terms of $N(0)$. This can be easily done if we iterate the matrix equation $N(t+1) = N(t) + Q$, where Q is as given in (21). In doing so, we get:

$$N(t) = \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} + \sum_{i=0}^t \begin{bmatrix} \mathbb{1}_{\{p^{(1)}(i)=1\}} & \dots & \mathbb{1}_{\{p^{(K)}(i)=1\}} \\ \mathbb{1}_{\{p^{(1)}(i)=2\}} & \dots & \mathbb{1}_{\{p^{(K)}(i)=2\}} \\ \vdots & \ddots & \vdots \\ \mathbb{1}_{\{p^{(1)}(i)=|V|\}} & \dots & \mathbb{1}_{\{p^{(K)}(i)=|V|\}} \end{bmatrix}. \quad (38)$$

Since this process is stochastic, it is worth determining the expectation of the number of visits $N(t)$ given the

agent's initial location $p(0)$. Noting that $\mathbb{E}[\mathbb{1}_{\{A\}}] = P(A)$, we have:

$$\begin{aligned} \mathbb{E}[N(t) | p(0)] &= \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \\ &+ \sum_{i=0}^t \begin{bmatrix} P^i(p_1(0), 1) & \dots & P^i(p_K(0), 1) \\ P^i(p_1(0), 2) & \dots & P^i(p_K(0), 2) \\ \vdots & \ddots & \vdots \\ P^i(p_1(0), |V|) & \dots & P^i(p_K(0), |V|) \end{bmatrix}, \end{aligned} \quad (39)$$

where $P^i(p_j(0), 1)$ denotes the $(p_j(0), 1)$ -entry of $\mathbb{P}_{\text{transition}}^i$. But, from the Markov Chains Theory, we have that the so-called truncated potential matrix [4] is given by:

$$R_t(v, k) \triangleq \sum_{i=0}^t \mathbb{P}_{\text{transition}}^i(v, k). \quad (40)$$

By virtue of (40), each entry of the matrix equation in (39) can be rewritten as:

$$\mathbb{E}[N_i^{(j)}(t) | p(0)] = 1 + R_t(p_j(0), i). \quad (41)$$

From (41), we can infer that each agent does perform an independent random walk according to a Markov Chain. Thus, we are able to conclude that, for $\lambda = 0$ and $\Delta = 0$, all the states of system ϕ follow a traditional Markov Chain process, except for a constant, as demonstrated in (41). \square

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

4 Application: Handwritten Digits and Letters Recognition

In this section, we provide an application of data clustering for the competitive method for three real-world data sets. Specifically, in Sect. 4.1, we derive a dissimilarity measure that we will use in the network formation step; in Sect. 4.2, we supply a brief description of the data sets; in Sect. 4.3, we show a method for determining the optimal number of

agents to be inserted into the network; and Sect. 4.4 reveals the data clustering results that we have obtained from our algorithm applied to the USPS, MNIST, and Letter Recognition data sets.

It is worth noting that in all simulations in this section, since the parameters ω_{\min} , ω_{\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 $\omega_{\max} = 1$.

4.1 The Network Formation Technique

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

In order to construct the network, we are required to establish a similarity measure. The traditional pixel-per-pixel distance is rather insufficient in terms of reliably representing data, since such measure is very sensitive to rotations and scale modifications. With the purpose of overcoming this difficulty, we propose a measure based on the eigenvalues that each image inherently carries with it. First of all, we remove the mean associated to each data item (image), so that we have a common basis of comparison. After that, we calculate the ϕ greatest eigenvalues of the image. Efficient methods have been developed for finding the leading eigenvalues of real-valued asymmetric matrices [12, 35]. The magnitudes of the eigenvalues are related to the variations that the image possesses; hence, it is a natural carrier of information [18]. The greater its value, more information about the image it conveys. By virtue of that, a good choice is to only extract the greatest $\phi < \eta$ eigenvalues and drop the smaller values, since these do not transport too much information about the image. Also, in order to give more emphasis to the largest eigenvalues, a weight is associated to each one so that the larger an eigenvalue is, the larger will be its associated weight.

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

similarity $1 - \rho$) between image i and j is given by:

$$\rho(i, j) = \frac{1}{\rho_{\max}} \sum_{k=1}^{\phi} \beta(k) [|\lambda_i^{(k)}| - |\lambda_j^{(k)}|]^2, \quad (42)$$

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

4.2 Brief Information of the Handwritten Digit and Letter Data Sets

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

- **USPS Data Set:** Comprised of 9298 images of handwritten digits. The digits 0 to 9 have 1553, 1269, 929, 824, 852, 716, 834, 792, 708, and 821 samples respectively. The USPS digits data were gathered at the Center of Excellence in Document Analysis and Recognition (CEDAR) at SUNY Buffalo, as part of a project sponsored by the US postal Service. For more details about this data set, refer to [14]. Each image has dimensions of 16×16 pixels, with 256 grey levels per pixel. We will employ the weighted eigenvalue similarity measure as studied before. Instead of using 16 eigenvalues, we will only work with the 4 greatest ones. In this case, the β function, as shown in (42), will be fixed as an exponential decreasing function with a time constant fixed at $\tau = 3$ and a scaling factor given by 16, i.e., $\beta(x) = 16 \exp(\frac{x}{3})$. Since, this function is mapped into the interval $(0, \infty)$ and is a monotonically decreasing function, it follows that this β function meets all the aforementioned requirements. Specifically in this situation, we have that the weights associated to each eigenvalue are: $\beta(1) = 11.46$, $\beta(2) = 8.21$, $\beta(3) = 5.89$, and $\beta(4) = 4.22$.
- **MNIST Data Set:** Originally composed of images with dimensions 28×28 . We will only use the public set composed of 10 000 vertices. Moreover, we will make use of the dissimilarity measure based on the first 4 eigenvalues of each image out of 28 eigenvalues. The same β function employed in the USPS data set will be used here.
- **Letter Recognition Data Set:** Composed of characteristic vectors with 16 entries. There are 20000 vertices.

Since none of these data sets are in a networked form, the methodology is divided into two general steps: the network formation and the data clustering tasks. In the first, we use the k -nearest neighbor network formation technique with $k = 3$ after we apply a pre-processing step. In this pre-processing, we standardize the data such as to have zero mean and unitary standard deviation. As for the distance measure, we either use the proposed weighted eigenvalue dissimilarity (for the first two data sets above) or the reciprocal of the Euclidean similarity (for the last one). The reason

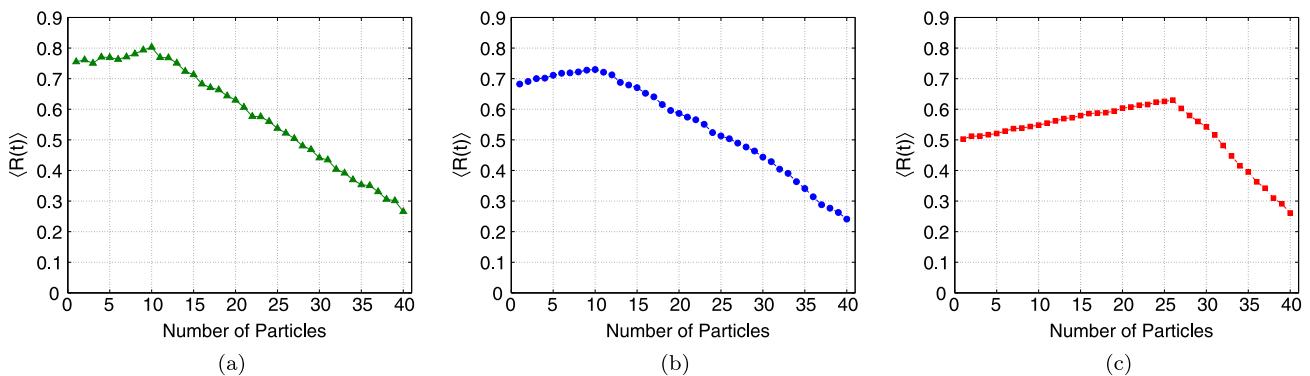


Fig. 1 Determination of the optimal number of agents K (the optimal number of clusters) in real-world data sets. The number of classes that each data set originally possesses are: (a) The USPS data set has 10 clusters (each cluster corresponding to a number from “0” to “9”); (b) The MNIST data set has 10 clusters (each cluster corresponding

to a number from “0” to “9”); and (c) The Letter Recognition data set has 26 clusters (each cluster corresponding to a letter from the English alphabet (“A” to “Z”). 20 independent runs are performed and the average value is reported

behind not using the weighted eigenvalue on the third data set is because the samples are not provided as images, but as image descriptors. Since the latter is formed by merely scalars, we cannot apply the proposed dissimilarity measure. In the second step, the data clustering algorithm based on particle competition is applied. As we are dealing with unsupervised learning, we do not use any external information, such as labels or exogenous knowledge. Instead, we limit ourselves to discovering explicit or implicit relationships among the data by the particle competition mechanism.

4.3 Determining the Optimal Number of Agents and Clusters

Remembering that our technique is sensitive to the parameter K , we need to identify the correct number of agents. In order to overcome this problem, we utilize the method proposed in [33] for determining the optimal number of agents to be inserted into the network. Such a method makes use of a useful network measure denominated average maximum domination level $\langle R(t) \rangle$ as follows:

$$\langle R(t) \rangle = \frac{1}{V} \sum_{u=1}^V \max_{m \in \mathcal{K}} (\bar{N}_u^{(m)}(t)), \quad (43)$$

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 other words, the maximum domination level of any vertex is a number near 1, showing that this vertex is being completely dominated by a single agent. 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 in the network. In

other words, the domination levels imposed by each agent in any arbitrary vertex are similar. 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.

Figures 1(a), 1(b), and 1(c) show the determination of the optimal K for the USPS, MNIST, and Letter Recognition data sets, respectively. One can verify that $\langle R(t) \rangle$ is maximized exactly when the number of agents is equal to the number of clusters in the network, confirming the effectiveness of such heuristic.

4.4 Handwritten Data Clustering

Here, we report the cluster detection accuracy reached by our algorithm in detail, along with the data clustering accuracy of a selected set of competing techniques. For the calculation of the cluster detection accuracy, we set that the ideal result is that each cluster represents a “digit” (in the USPS and MNIST data set) or a “letter” (in the Letter Recognition data set). Particularly, Table 1 supplies details about the algorithms that we have chosen for comparison matters. We have used the genetic algorithm available in the Global Optimization Toolbox of MATLAB with its default parameters in order to optimize the parameters of our algorithm. In our

Table 1 Description of the competing state-of-art data clustering techniques.

Technique	Reference
Gaussian Mixture Model (GMM)	[3]
K-Means	[22]
Locally Consistent Gaussian Mixture Model (LCGMM)	[21]
Spectral clustering algorithm with normalized cut (Ncut)	[31]
Ncut Embedding All (NcutEmb ^{All})	[29]
Ncut Embedding Maximum (NcutEmb ^{Max})	[29]

Table 2 Data clustering accuracy reached by the proposed technique and the competing methods listed in Table 1. For the stochastic methods, such as the particle competition method, thirty independent runs were performed and the corresponding mean is provided

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

case, we have optimized λ over the range $0.2 \leq \lambda \leq 0.8$. This optimization process is conducted in a way to choose the best λ for which the data accuracy reaches its maximum value. By doing this, we get that the optimal values of λ for the USPS, MNIST, and Letter Recognition data sets to be 0.58, 0.60, 0.60, respectively. The number of agents to be inserted is determined accordingly to the previously analysis that we have shown, i.e., with the aid of the $\langle R(t) \rangle$ measure. We choose the number of agents that maximizes this quantity. By looking at Figs. 1(a) (USPS), 1(b) (MNIST), and 1(c) (Letter Recognition), we are able to conclude that the maximum value of $\langle R(t) \rangle$ is attained when the number of agents is 10, 10, and 26, respectively.

Table 2 reports the data clustering accuracy reached by our method and the aforementioned competing algorithms. Some of these results are readily extracted from [29] and [21]. For more information about the parameters used in the competing techniques, see the aforementioned references. Within this table, we have provided the Average Rank of each algorithm, which is calculated as follows: (i) for each data set we rank the algorithms according to their average performance (average data clustering accuracy), i.e., the best algorithm is ranked as 1, the second best one is ranked as 2, and so on; (ii) for each algorithm, the Average Rank is given by the average value of its rank achieved in all the data sets. As we can verify by looking at the Average Rank column, our algorithm has reached one of the best positions, showing the effectiveness of the proposed technique. In order to examine the results in a statistical manner, we utilize an adaptation of [7] for clustering matters. The methodology described therein assigns a rank to each algorithm on each data set according to its final accuracy reached. After this step, the average rank of each algorithm is then evaluated and a Friedman Test is applied to the resulting average rank values of each algorithm. The Friedman Test is used to check whether the measured average ranks are significantly distinct from the mean rank, in this case 4.0, because there are seven data clustering techniques. The null-hypothesis considered here is that all the

algorithms are equivalent, so their ranks should be equal. Hereon, for all the future tests, we fix a significance level of 0.10. For our experiments, according to [7], we have that $N = 3$ and $k = 7$, resulting in a critical value given by $F(6, 12) = 2.33$, where the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. In our case, we get a value $F_F \approx 4.00$ that is higher than the critical value, so the null-hypothesis is rejected at a 10% significance level. Thus, we conclude that the algorithms at hand present statistical difference, i.e., they are statistically different from the mean rank. Nonetheless, one can see that our algorithm has obtained the best average rank in relation to the other algorithms for these three data sets.

As the null hypothesis is rejected, we are able to advance to post-hoc tests which aim at verifying the performance of our algorithm in relation to others. For this task, we opt to use the Bonferroni-Dunn Test, with the proposed technique fixed as the control algorithm. According to [7], one should not make pairwise comparisons when we test whether a specific method is better than others. Basically, the Bonferroni-Dunn Test quantifies whether the performance between an arbitrary algorithm and the reference is significantly different. This is done by verifying whether the corresponding average ranks of these algorithms differ by at least a critical difference (CD). If they do differ that much, then it is said that the better ranked algorithm is statistically superior to the worse ranked one. Otherwise, they do not present a significant difference for the problem at hands. Thus, if we perform the evaluation of the CD for our problem, we encounter $CD = 4.22$ when the significance level is 10%. The average rank of the proposed method is 2.00. By virtue of that, if any rank does lie in the interval 2.00 ± 4.22 , the control algorithm and the compared algorithms are statistically equivalent. We conclude that our algorithm is superior to K-Means for this set of databases. However, the other pairwise comparisons to the control algorithm do not surpass the CD, meaning that the differences among them are statistically insignificant. Nonetheless, the proposed technique has obtained the best performance (the best average rank) in relation to the other techniques and also presents reasonable computational complexity as we have previously addressed.

In order to further verify the robustness of the proposed technique, we inspect the samples that compose a same cluster. Specifically, Figs. 2, 3, 4, and 5 show some samples of the clusters representing the pattern “2”, “5”, “6”, and “8”, respectively, of the MNIST data set. These samples are captured using the following strategy: we compute the vertices that compose the maximum geodesic distance of the cluster representing each pattern (cluster diameter). Now, we select a representative subset of vertices composing the cluster diameter trajectory for illustrative purposes. In these figures, samples that are adjacent are more similar than those distant

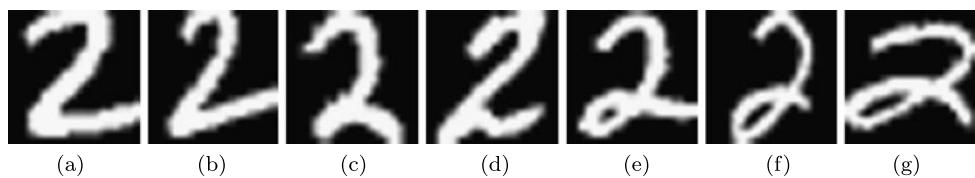


Fig. 2 A broad set of samples of that were classified as being member of the cluster representing the pattern “2”. Note that samples that are adjacent are similar with regards to the weighted eigenvalue dissimilarity function. The transitions from the sample **(a)** to **(g)** were

captured from the maximum geodesic distance between two vertices in the cluster representing pattern 2. In this case, the diameter of such cluster is 17. We have only provided 7 representative samples above

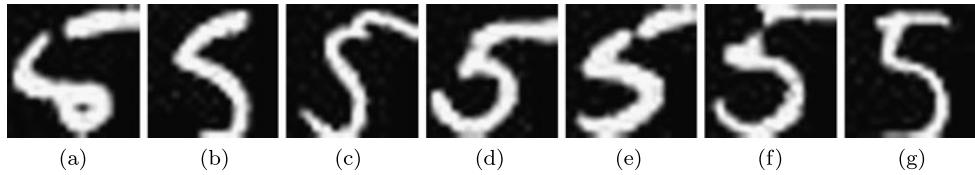


Fig. 3 A broad set of samples of that were classified as being member of the cluster representing the pattern “5”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples. The methodology for construction (geodesic path) is the same

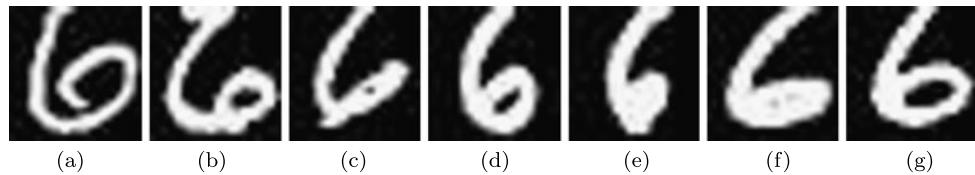


Fig. 4 A broad set of samples of that were classified as being member of the cluster representing the pattern “6”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples. The methodology for construction (geodesic path) is the same

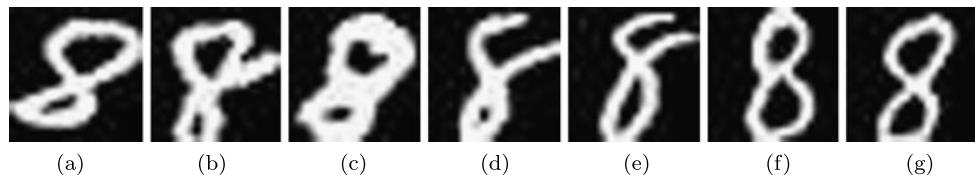


Fig. 5 A broad set of samples of that were classified as being member of the cluster representing the pattern “8”. Likewise the previous figure, adjacent samples are more similar to each other than distance samples. The methodology for construction (geodesic path) is the same

from one to another. On the basis of this analysis, we conclude that the graph representation has successfully captured several variations of the these number patterns each of which in a single representative cluster, showing the robustness of the proposed model.

5 Conclusions

This paper studies a new model for competitive learning in networks, biologically inspired by the competition process taking place in many nature and social systems. In this model, several agents navigate in the network to explore their territory and, at the same time, attempt to defend their territory from rival agents. If an agent frequently visits a spe-

cific vertex, it occurs that the domination level of the visiting agent to this vertex is strengthened; simultaneously, the domination levels of all the other agents on the same vertex are weakened. Finally, each agent is confined in a community of the network.

The competitive model is nonlinear and stochastic. In addition, we show that it generalizes the process of multiple independent random walks when a special case occurs. Such generalization is realized by changing the values of the parameters λ and Δ of the system. If $\lambda = 0$ and $\Delta = 0$, the proposed model reduces to multiple independent random walks. Conversely, when $\lambda > 0$, the competition mechanism is turned on. In this case, the reanimation procedure is enabled depending on the parameter Δ . Moreover, here we

provide a mathematical analysis of the unsupervised competitive model.

Simulations are carried out with the purpose of quantifying the robustness of the proposed technique on handwritten data clustering. Computer simulations show that the competitive model works well for detecting digits or letters even with considerable distortion in the same cluster.

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

References

1. Arenas, A., Diáz-Guilera, A., Pérez-Vicente, C.J.: Synchronization reveals topological scales in complex networks. *Phys. Rev. Lett.* **96**(11), 114102 (2006)
2. Berkhin, P.: Survey of clustering data mining techniques. *Tech. rep., Accrue Software* (2002)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. *Information Science and Statistics*. Springer, New York (2006)
4. Çinlar, E.: Introduction to Stochastic Processes. Prentice-Hall, Englewood Cliffs (1975)
5. Cinque, L., Foresti, G., Lombardi, L.: A clustering fuzzy approach for image segmentation. *Pattern Recognit.* **37**, 1797–1807 (2004)
6. Danon, L., Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. *J. Stat. Mech. P09008* (2005)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Unsupervised learning and clustering. In: *Pattern Classification*, 2nd edn. Wiley-Interscience, New York (2001)
9. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010)
10. Fortunato, S., Latora, V., Marchiori, M.: Method to find community structures based on information centrality. *Phys. Rev. E* **70**(5), 056104 (2004)
11. Gan, G.: Data Clustering: Theory, Algorithms, and Applications vol. 20. SIAM, Philadelphia (2007)
12. Goldhirsch, L., Orszag, S.A., Maulik, B.K.: An efficient method for computing leading eigenvalues and eigenvectors of large asymmetric matrices. *J. Sci. Comput.* **2**, 33–58 (1987)
13. Govindan, V., Shivaprasad, A.: Character recognition: a review. *Pattern Recognit.* **23**, 671–683 (1990)
14. Hull, J.J.: A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.* **16**, 550–554 (1994)
15. Husek, D., Pokorný, J., Rezankova, H., Snasel, V.: Data clustering: from documents to the web. In: *Web Data Management Practices: Emerging Techniques and Technologies*, pp. 1–33. IGI Global, Hershey (2006)
16. Jain, A.K.: Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **31**, 651–666 (2010)
17. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. *ACM Comput. Surv.* **31**(3), 264–323 (1999)
18. Jolliffe, I.T.: In: *Principal Component Analysis*, 1 edn. Springer Series in Statistics (2002)
19. Karypis, G., Han, E.H., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. *IEEE Computer* **32**(8), 68–75 (1999)
20. Liu, C.L., Sako, H., Fujisawa, H.: Performance evaluation of pattern classifiers for handwritten character recognition. *Int. J. Doc. Anal. Recognit.* **4**, 191–204 (2002)
21. Liu, J., Cai, D., He, X.: Gaussian mixture model with local consistency. In: *AAAI'10*, vol. 1, pp. 512–517 (2010)
22. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
23. Mori, S., Suen, C., Kamamoto, K.: Historical review of ocr research and development. *Proc. IEEE* **80**, 1029–1058 (1992)
24. Newman, M.: Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **69**(6), 066133 (2004)
25. Newman, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2), 026113 (2004)
26. Newman, M.: Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**(3), 036104 (2006)
27. Pradeep, J., Srinivasan, E., Himavathi, S.: Diagonal based feature extraction for handwritten alphabets recognition system using neural network. *Int. J. Comput. Sci. Inf. Technol.* **3**, 27–38 (2011)
28. Quiles, M.G., Zhao, L., Alonso, R.L., Romero, R.A.F.: Particle competition for complex network community detection. *Chaos* **18**(3), 033107 (2008). doi:[10.1063/1.2956982](https://doi.org/10.1063/1.2956982)
29. Ratle, F., Weston, J., Miller, M.L.: Large-scale clustering through functional embedding. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases—Part II*, ECML PKDD'08, pp. 266–281. Springer, Berlin (2008)
30. Reichardt, J., Bornholdt, S.: Detecting fuzzy community structures in complex networks with a potts model. *Phys. Rev. Lett.* **93**(21), 218701 (2004)
31. Shi, J., Malik, J.: Normalized cut and image segmentation. *Tech. rep., Berkeley, CA, USA* (1997)
32. Silva, T.C., Zhao, L.: Stochastic competitive learning applied to handwritten digit and letter clustering. In: *XXIV SIBGRAPI: Conference on Graphics, Patterns and Images*, pp. 313–320 (2011)
33. Silva, T.C., Zhao, L.: Stochastic competitive learning in complex networks. *IEEE Trans. Neural Networks Learn. Syst.* **23**(3), 385–398 (2012)
34. Theodoridis, S., Koutroumbas, K.: *Pattern Recognition*, 4th edn. Academic Press, New York (2008)
35. Tsai, S.H., Lee, C.Y., Wu, Y.K.: Efficient calculation of critical eigenvalues in large power systems using the real variant of the Jacobi-Davidson qr method. *IET Gener. Transm. Distrib.* **4**, 467–478 (2010)
36. Zhou, H.: Distance, dissimilarity index, and network community structure. *Phys. Rev. E* **67**(6), 061901 (2003)